Luís Tiago Flores Cristóvão

Cidades Inteligentes - Um Jogo Digital Sério

Dissertação para obtenção do Grau de Mestre em
Engenharia Eletrotécnica e Computadores

Orientador: Tiago Cardoso, Faculdade de Ciências e Tecnologia

Júri:

Presidente: Prof. Doutor Rodolfo Alexandre Duarte Oliveira

Arguente: Prof. Doutor João Almeida das Rosas

Vogal: Prof. Doutor Tiago Oliveira Machado de Figueiredo Cardoso

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2017

Cidades Inteligentes - Um Jogo Digital Sério

Agradecimentos

Gostaria de agradecer, ao meu orientador Professor Tiago Cardoso pela oportunidade que me concedeu de trabalhar neste tema, pelo apoio disponibilidade demonstrada ao longo da construção desta dissertação. Queria também agradecer aos meus pais, família e amigos, pela paciência e apoio. Também agradeço todos os que jogaram e responderam ao questionário que permitido a realização desta dissertação. Por fim agradeço a Deus.

Resumo

As cidades são grandes centros de desenvolvimento e de empregabilidade, que atrai cada vez mais pessoas. O número de pessoas a morar nas cidades aumentou, mas os recursos das cidades não estão a aumentar, por isso é necessário não só produzir mais recursos, como fazer uma gestão mais inteligente dos mesmos. Com os avanços das tecnologias de informação e comunicação (TIC), tornou-se possível visualizar melhor o consumo de recursos que uma cidade faz, assim eliminando desperdícios. Desta forma, nasceu um conceito que tem crescido em popularidade designado de Cidades Inteligentes. No fundo é usar os novos avanços tecnológicos para fazer uma gestão mais inteligente dos recursos da cidade.

Tendo em conta que as cidades inteligentes são o futuro das cidades, considerou-se relevante ensinar este conceito aos mais jovens. Sabendo que os jovens gostam muito de jogos, e que cada vez mais, os jogos são usados como ferramenta de aprendizagem, considerou-se realizar um jogo educativo (jogo sério) que ensine este conceito aos mesmos.

O objetivo desta dissertação, é estudar a performance dos jogos digitais a ensinar os conceitos básicos e fundamentais, sobre as Cidades Inteligentes. A validação deste trabalho consistiu no desenvolvimento de um jogo protótipo, que incentive os jogadores a aprenderem enquanto jogam. O jogo protótipo consta na construção e gestão de uma cidade inteligente, onde o jogador é que toma todas as decisões para tornar a cidade sustentável. Para avaliar o protótipo desenvolvido, pretende-se recolher dados sobre a forma de como os jogadores interagiram com o jogo. Por fim, será pedido a realização de um questionário, que deve ser respondido pelos jogadores, para validação do desempenho do jogo, tanto na sua componente educativa como lúdica.

Palavras-chave: Cidades Inteligentes, Jogos Sérios, Jogo Educacional, Simulador de Cidade.

Abstract

Cities are great centers of development and employability, which attracts more and more people to live in it. The abundance of people in cities is a problem, because their resources are limited, so it is necessary not only to produce more, but also do a better management of them. With the advancements in information and communication technologies (ICT), it is possible to better visualize the consumption of city resources, thus eliminating waste. This way, it was born a concept that is growing in popularity, which is called Smart Cities. In short, smart cities take advance of the new ICT to make a smarter management of the city resources.

Given that smart cities are the future of cities, it was considered relevant to teach this concept to the younger generation. Knowing that young people are very fond of games, and that more and more games are used as a learning tool, it was considered to carry out an educational game (serious game) that teaches this concept to young people.

The purpose of this dissertation is to study the performance of digital games, to teach the basic and essential concepts of Smart Cities. The validation of this work consisted of the development of a prototype game that encourages players to learn while playing. The prototype game consists of building and managing a smart city, where the player makes all the decisions to make the city sustainable. To know whether the prototype worked, it is intended to collect information about the players, on how they played the game. Finally, a questionnaire will be asked to do, by the players, to validate the performance of the game in both its educational and playful component.

Keywords: Smart Cities, Serious Games, Educational Games, City Simulator.

CONTEÚDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

1 INTRODUÇÃO

O objetivo desta dissertação é estudar a capacidade que têm os jogos digitais, para ensinar as crianças e jovens, conceitos sobre Cidades Inteligentes ou Smart Cities (**SC**) como são mais conhecidas. Os jogos digitais aplicados à educação não é algo novo, desde "Pac-Man" em 1980, que educadores querem usar o entusiasmo ligado aos jogos digitais como forma de envolver e ensinar [1]. De todas as formas de entretenimento que existem, os jogos têm algo único e relevante para esta área da educação, que é a interação que existe entre a pessoa e o jogo. Descobriu-se que através da audição de aulas/palestras, indivíduos só retêm 5% da matéria dada enquanto que por experimentação (realização de tarefas), retêm 75% [2]. Este fator na aprendizagem de conhecimentos, torna os jogos digitais numa ferramenta muito atrativa para educadores e alunos. Simuladores de condução e aviação, já existem há algum tempo, e provam como é benéfico o uso de jogos para ter um primeiro contacto, com um problema, num ambiente protegido que possibilite o erro e a aprendizagem. Com os avanços tecnológicos, os jogos tornaram-se cada vez melhores, fazendo com que a sua popularidade crescesse muito, principalmente na população jovem [2]. Por isso achou-se interessante usar este recurso, dos jogos digitais, principalmente na área dos jogos sérios, para ensinar sobre o tópico das **SC**.

O número de pessoas que moram nas cidades tem crescido todos os anos, isto deve-se ao facto de as cidades serem grandes centros de empregabilidade e de bens, que atraem cada vez mais pessoas[3]–[5]. Este crescimento provoca grandes problemas na gestão dos recursos da cidade[6], tornando a gestão atual limitada para o grande conjunto de pessoas que moram nas cidades. Portanto é necessário não só produzir mais, como fazer uma gestão mais inteligente e automática dos seus recursos. Os avanços tecnológicos, principalmente na área de tecnologias de informação e comunicação (TIC ou ICT) e não só, são vistas como algumas das medidas que podem resolver este problema, de tornar a gestão de recursos das cidades mais inteligente. Devido a estes fatores, as **SC** são cada vez mais vistas, como o futuro das Cidades. Como as **SC** são o futuro das cidades e as crianças e jovens são o nosso futuro, achou-se interessante realizar esta dissertação.

1.1 JOGOS SÉRIOS

O que são jogos sérios? quais as diferenças entre jogos "normais" e jogos sérios? São questões a abordar neste capítulo. Um jogo sério é aquele que tem como objetivo principal a educação e não o entretenimento[7], apesar de estar incluído.

A maior diferença entre videojogos e jogos sérios é o seu propósito[8]. "Os videojogos, são desenhados desde o início para o entretenimento, sem ter a intenção direta de ensinar"[8]. Os jogos sérios são desenhados desde o começo para ter em conta a componente séria (educacional) e a componente de jogo (entretenimento)[8]. A combinação entre estas duas dimensões, pedagógicas e lúdicas é a principal dificuldade na criação de um jogo sério[8].

É preciso ter em conta a seguinte distinção entre jogos exclusivos ao entretenimento e os jogos sérios. Os jogos digitais apesar de serem criados com o propósito de entretenimento, sempre podem servir propósitos educativos, mas isto não o torna num jogo sério, visto que não foi desenhado, construído com esse propósito, apenas tem componente pedagógica se algum fator externo, (por exemplo uma pessoa) o direcionar para tal. O próximo parágrafo é um citação da referência [9].

Um exemplo disto é a comparação entre dois jogos "Trauma Center: Under the Knife" e "Pulse!!" sendo o primeiro apenas um jogo digital enquanto que o segundo é um jogo sério[9]. "Por exemplo, o jogo Trauma Center: Under the Knife lança jogadores como cirurgiões e pede-lhes para operar em pacientes. No entanto, além do tema do hospital e certas referências ao equipamento e técnicas cirúrgicas da vida real, o jogo não foi projetado com um cenário explícito de "propósito sério""[9]. "Neste jogo, a saúde é usada apenas como um pano de fundo para construir um cenário divertido para o jogo"[9]. Por outro lado, Pulse!! é muito diferente, mesmo que os jogadores também tenham como papel, ser médicos. "Os designers do Pulse !! introduziram um cenário de treino médico no cenário do jogo, a fim de fornecer um propósito sério para o jogo"[9]. "As diferenças são ainda mais evidentes ao jogar estes dois jogos. Enquanto "Trauma Center: Under the Knife" pede aos jogadores que usem um laser no coração de seus pacientes para matar um vírus em forma de dragão, Pulse !! fornece-lhes casos reais que têm de ser resolvidos usando técnicas médicas atuais"[9].

Portanto os jogos sérios têm propósitos bastante diferentes em relação aos videojogos "normais". Segundo [9] os propósitos dos jogos sérios, podem-se resumir em 3 componentes: **transmissão de mensagens**, **treino** e **troca de dados.**

Transmissão de mensagens: como o nome indica, é um jogo desenhado com o propósito de transmitir uma mensagem. Todos os jogos educacionais incluem-se nesta secção pois estão a transmitir algum tipo de conhecimento.

Treino: nem todos os jogos sérios servem para transmitir conhecimento ou mensagens, alguns também servem para treinar pessoas a realizar tarefas, que de outro modo, seria difícil de treinar num ambiente seguro. Por exemplo todos os jogos do tipo simulação servem para treinar pessoas, desde simuladores de condução a jogos para treino militar, todos eles servem para treinar um exercício uma tarefa antes de a realizar.

Troca de dados: este último componente serve por exemplo, para haver troca de dados entre jogadores e desenvolvedores ou entre os próprios jogadores. Alguns investigadores para fazer estudos hoje em dia, preferem realizar estudos através de vídeo jogos. Em vez de o método tradicional de realizar questionários, que podem captar menos informação sobre a pessoa (caso esta minta no formulário por exemplo), podem optar por realizar um jogo que capte de forma discreta e divertida, informações sobre o jogador. Também existem jogos onde os jogadores são incentivados a partilhar informação entre eles, aprendendo mais partilhando informação e conhecimento.

1.1.1 Mercado e Publico Alvo

Enquanto o mercado principal dos vídeo jogos é o entretenimento, os mercados onde os jogos sérios se podem inserir são muitos mais variados, como: Estado & Governo, Militar & Defesa, Saúde, Educação, Empresarial, Religioso, Cultural & Artístico, Ecologia, Politica,

Humanitária, Publicidade e Investigação Cientifica [9]. O próprio mercado do entretenimento é uma possibilidade para os jogos sérios visto que também contêm uma componente de entretenimento, por exemplo o jogo America's Army é usado tanto em treino militar como em torneios de vídeo jogos [9].

1.1.2 Desenvolvimento

O desenvolvimento de jogos sérios tem semelhanças e diferenças em relação aos videojogos para o entretenimento, este capítulo pretende discutir esse assunto. Em primeiro lugar uma grande diferença no processo de desenvolvimento de ambos, é no levantamento de requisitos[9]. Por exemplo, para produzir um jogo sério para profissionais de Saúde é necessário reunir com os mesmos e perceber o que pretendem treinar ou aprender com o jogo, portanto não só é necessário recolher os requisitos do jogo com o cliente, mas é também necessário recolher informação sobre a área de conhecimento específica, para onde se está a produzir o jogo. Nos jogos virados para o entretenimento, isto não acontece.

Depois de levantar os requisitos é necessário formular o cenário do jogo sério [9], que vai necessitar de duas pessoas, ou dois grupos diferentes de pessoas, um para criar o cenário pedagógico e outro para desenhar o cenário lúdico ou de jogo (exemplo: puzzle, história, etc…). As pessoas responsáveis por estes dois cenários terão de juntar-se e discutir uma forma de juntar os dois cenários. Depois de formular o modelo do jogo é necessário desenvolvê-lo e testá-lo. Como sempre, tanto nos jogos lúdicos como os sérios, a parte de desenvolvimento e de teste estão bastante ligados, pois a partir dos testes nos jogadores é possível receber feedback para melhorar o jogo. A diferença está no objetivo, pois para testar os jogos sérios é necessário testá-los em duas áreas, a componente pedagógica e a de entretenimento, pois se um jogo sério é muito bom a entreter, mas não ensina, não cumpre o objetivo para o qual foi criado, caso ensine, mas seja muito aborrecido, então perde o valor de ser um videojogo, tornando preferível recursos alternativos de ensino.

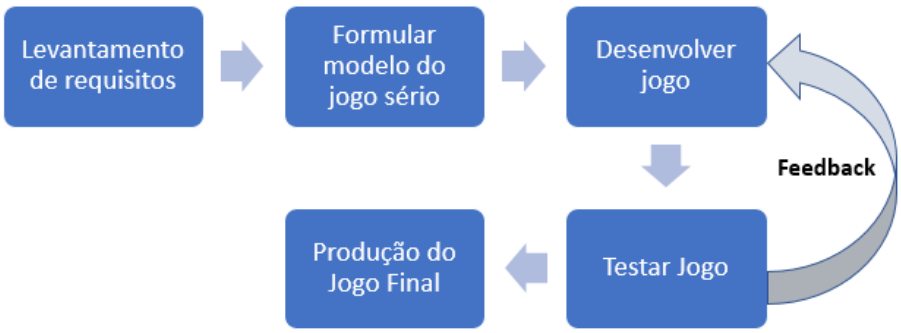


Figura 1 - Ciclo de desenvolvimento de jogos sérios

Na Figura 1, é possível visualizar graficamente o ciclo de desenvolvimento de um jogo sério. Outra coisa a considerar, é a diferença no risco económico. Porque apesar de os jogos sérios terem mais possíveis mercados para produzir jogos, esses mercados muitas vezes têm menos clientes do que o mercado do entretenimento, por exemplo se se realizar um jogo sério para ajudar crianças com um problema médico raro, o jogo terá muito menos clientes do que o mercado tradicional dos jogos, o de entretenimento. Por outro lado, porque os jogos sérios tentam juntar duas áreas, muitas vezes consideradas como opostas, torna-se muito difícil fazer

um “bom” jogo sério que junte bem a componente lúdica com a educacional, fazendo assim com que o sucesso de jogos sérios seja ainda menor.

1.2 Estrutura do Documento

O documento está dividido em cinco capítulos distintos, sendo a sua estrutura identificada em seguida:

• **Capítulo 2**: será apresentado uma breve análise do estado de arte sobre as Cidades Inteligentes. De seguida serão apresentados alguns jogos encontrados, que sejam semelhantes ao que esta dissertação pretende realizar. Por fim, neste capítulo, será discutido qual a melhor ferramenta para criação do jogo.

• **Capítulo 3**: pretende explicar a solução encontrada, apresentando de uma forma conceptual o modelo encontrado para a satisfação dos requisitos funcionais identificados.

• **Capítulo 4**: pretende explicar o jogo construído e validar os requisitos proposto no capítulo 3, através da realização de testes e consequente análise de resultados.

• **Capítulo 5**: capítulo final onde se apresentam as conclusões retiradas sobre o trabalho efetuado, bem como uma visão futura para possíveis desenvolvimentos deste trabalho.

2 ESTADO DE ARTE

Tendo como objetivo da Dissertação a implementação de um jogo sério sobre **Smart Cities** ("Cidades Inteligentes"), este capítulo será dedicado aos seguintes temas: o que **são "Smart Cities" (SC), quais os seus conceitos fundamentais**, **que jogos já existem sobre tema e por fim que plataforma ou ferramenta usar para realizar o jogo**. Considera-se relevante abordar os temas previamente enunciados, visto que para implementar um jogo sobre **SC**, é necessário recolher o máximo de informação possível, sobre o tema (o estado de arte) e para não se fazer um jogo igual aos que já existam no mercado é importante saber o que já existe.

2.1 SMART CITIES

Sendo assim, comecemos primeiro pelas **Smart Cities**(**SC**). O conceito das **SC** apareceu para tentar resolver o problema de aumento de população nas cidades [3]–[6], [10], que faz com que a procura de recursos aumente muito, e coloque pressão nos serviços que as cidades dispõem [6]. Quando olhamos para as cidades a última coisa que podemos dizer é que são "inteligentes", pois o que ressalta à vista é a quantidade de tráfego nas cidades, os excessos e desperdícios [11] de comida, água e energia, etc. A poluição nas cidades é elevadíssima ao ponto de chegar a matar [12] e a produção de lixo nas cidades é muito alta, sendo 1.2 kg por pessoa todos os dias [13]. Podemos perceber que a gestão das cidades como estão, hoje em dia, está no limiar de uma rotura, precisamos de uma gestão mais inteligente dos recursos! A situação só vai piorar à medida que o número de pessoas nas cidades aumentem [4], [5]. Sendo o conceito das **SC** visto como a solução deste problema, como defini-la? Não existe uma definição absoluta, mas de acordo com [14] pode ser considerado como **uma série de passos, que torne a cidade mais habitável e resiliente**. Também de acordo com a mesma fonte, as **SC** devem tornar os **cidadãos no foco central da cidade**, fazendo-os mais participantes na cidade, tanto do lado do consumidor como de fornecedor de serviços. Segundo [10] a **SC** mistura tecnologia, governo e sociedade, para criar as seguintes características[10], [15], [16], do que se considera ser uma **SC**:

- **Smart economy**: este conceito de economia inteligente, visa usar as novas tecnologias de informação e comunicação (TIC ou ICT) para dinamizar e melhorar todos os assuntos relacionados com economia numa cidade, como comércio, emprego, negócios, bancos etc [15], [16].

- **Smart mobility**: ou mobilidade inteligente visa tornar os sistemas de transportes mais inteligentes, eficazes e otimizados. Não é sempre necessário haver uso de tecnologias para tornar os sistemas de transportes, mais inteligentes, mas claro que ajuda muito, por exemplo, existir mais incentivos para usar veículos não motorizados, pode ser uma medida de **smart mobility**. Também existem iniciativas onde o uso de tecnologia é mais óbvio, como ter dados de tráfego da cidade[17], para saber os melhores caminhos para um certo destino. O uso de transportes autónomos (exemplo comboios automáticos[18]), também pode ser visto como uma medida de **smart mobility**, que ajudaria, por exemplo a tornar os transportes públicos mais atrativos para o cidadão (assim reduzindo o tráfico nas cidades) fazendo com que o consumo de energia fosse mais racionalizado [15], [16].

- **Smart Environment**: “é a característica das Smart Cities que está diretamente relacionada com o desenvolvimento urbano sustentável e com a gestão responsável dos seus recursos. Desde iniciativas para promover a redução de consumos energéticos, implementação de políticas para redução de emissões poluentes a esforços para preservação e manutenção de espaços verdes dentro das cidades, à integração de novas tecnologias na gestão destas áreas revela-se de extrema importância”. Este parágrafo é uma citação de [15].

- **Smart people**: tem a haver mais uma vez, como o uso das TIC e não só, para tornar os cidadãos das cidades mais inteligentes, mais participativos na criação de uma cidade melhor. Para isso as cidades deverão disponibilizar fácil acesso a informação, sobre a cidade e não só, cursos online e “offline”, os cidadãos devem ter fácil acesso a tecnologia que possibilite criação de produtos, ou seja, facilite o empreendedorismo. Finalmente, deve também facilitar a procura de emprego e incentivar à inovação [15], [16].

- **Smart Living:** “é a característica das **SC** que está voltada para o bem-estar do cidadão. A integração de novas tecnologias de informação e comunicação no sistema de saúde, educação, segurança e serviços sociais permite aumentar a qualidade de vida do cidadão, na medida em que a informação e o acesso e estes serviços poderá, por exemplo, ser realizado comodamente através do computador de casa ou a partir de um Smart Phone”. Este parágrafo é uma citação de [15].

- **Smart governance**: “refere-se ao grau de interligação entre a administração da cidade e os seus cidadãos, instituições ou empresas. Através de infraestruturas de informação e comunicação os cidadãos podem aceder às informações relativas à gestão da sua cidade, o sistema político quer-se transparente por forma a que o cidadão tenha parte ativa nas tomadas de decisão. Esta transparência usualmente assenta em serviços de informação, como por exemplo o e-government, que facilita a comunicação entre as administrações municipais e as instituições e pessoas”. Este parágrafo é uma citação de [15].

  Enunciado os conceitos básicos sobre as **SC**, convém fazer um resumo do que foi dito. As Smart Cities são um conceito que veio responder ao problema da gestão das cidades com muitos habitantes. A solução que as **SC** trazem, não passa necessariamente pelo uso de tecnologia [19], [20], mas a tecnologia seria a estrutura base para realizar uma melhor gestão da cidade. Para gerir a cidade inteligente, primeiro é preciso extrair informação, dados, no fundo, **conhecimento** [11] sobre ela, de modo a poder “atacar” onde existe maior desperdícios. Por isso, aparece um conceito importante relacionado com as **SC** que é a internet das coisas, mais conhecido como “internet of things” (IoT), que visa interligar sensores e atuadores através da internet. Recolher informação é um fator crítico para a gestão, o que faz com que o IoT seja fundamental nas **SC**. Todo o tipo de tecnologia, ideias ou iniciativas que ajudem a otimizar a utilização de recursos na cidade e a melhorar o nível de vida global das cidades, faz parte da essência das **SC**.

  Tendo concluído o conceito, é necessário perguntar, o que já existe feito? que tecnologias ou iniciativas já se fizeram ou se pensa fazer no futuro das **SC**? A próxima

secção pretende expor as novas tecnologias, conceitos e iniciativas que já se fizeram no mundo, a propósito das **SC**, de modo a depois poder incluir alguma delas no jogo final.

2.2 Cidades, Tecnologias e Inovações Sobre as Smart Cities

2.2.1 Masdar

A cidade de **Masdar** é um dos melhores exemplos de Smart City pois foi construída desde o princípio com esse propósito. A cidade não permite o uso de carros para transporte na cidade, por isso não tem estradas para automóveis[21], só tem estradas pedonais ou para bicicletas[21]. Estas estradas são estreitas e pequenas, por isso todos os edifícios estão mais perto um dos outros, isto faz com que os edifícios produzam sombra na cidade, assim diminuindo a temperatura da cidade durante o dia. Pretende-se que os edifícios sejam inteligentes, ou seja, deverão usar tecnologia de monitorização de energia, água, etc…[22]–[24] de modo a fazer melhor uso dos recursos. Os edifícios foram desenhados de forma a não produzir resistência à passagem de vento (para tornar a cidade mais fresca) e toda a cidade está orientada para nordeste de modo a reduzir o tempo de exposição solar durante o dia[24]. Em termos de mobilidade, existe debaixo da cidade carros elétricos, automáticos (sem condutor)[15], [21], [25] onde os passageiros podem viajar pela cidade de forma, mais rápida. Também têm um metro de superfície que percorre a cidade e faz o transporte para cidades exteriores[21]. Por fim em termos de energia, a cidade funciona quase 100%[15], [21], [24]–[26] com energias renováveis tais como solar, geotérmico e desperdício[21], [24]. O consumo de água é um dos mais eficientes, tendo um dos menores consumos per capita do mundo[15], [24], reutilizando a água usada para regar os espaços verdes da cidade[21].

2.2.2 Transformação Digital De Serviços Governamentais

A transformação digital de serviços governamentais é cada vez mais uma realidade, visto que muitos deles podem ser utilizados e consultados online. Portugal é um bom exemplo, sites como portal do cidadão e portal das finanças[27], [28], são bons modelos disso, pois permitem tratar de finanças, emprego, cartão de cidadão, entre outros, tudo online. A Cidade de Estocolmo, é das mais avançadas, permitindo por exemplo aos pais a possibilidade de inscrever seus filhos em escolas, tudo através da internet, ou obter licenças ou permissões de construção ou de lugar de estacionamento, entre outros. [29], [30].

2.2.3 Energia

Em termos de inovações na área de energia nas cidades, podemos observar, que cada vez mais se está a apostar nas energias renováveis. A ideia de **smart grid** (“rede elétrica inteligente”) é um conceito revolucionário na área de energia, descentralizando a produção de energia, tornando-a barata para todos. Claro que ainda existe problemas na implementação desta tecnologia, mas a ideia é que cada casa possa produzir a sua própria energia, e caso não produza o suficiente para as suas necessidades, pode comprar mais à rede, se produzir em excesso pode vender à rede [31]–[34].

2.2.4 Inovações

Outras tecnologias que facilitam o dia a dia nas cidades, são por exemplo as que a empresa Amazon está a tentar implementar. **Amazon Go**![35] é um novo paradigma de compras de mercearia, onde não existe fila de espera, a pessoa apenas escolhe os produtos e saí da loja, e automaticamente o sistema sabe que produtos essa pessoa tirou e faz a compra de forma digital. O LIDL também já realizou algo do género, o LIDL SHOP&GO[36], onde uma pessoa pode ler através do telemóvel (smartphone) o código de barra dos produtos e o sistema coloca-os na sua lista de compras online, depois é só mostrar a lista (usando um QRcode) numa caixa automática e pagar, facilitando assim a compra, não passando por fila de espera. Tecnologias como os drones vão também revolucionar o funcionamento das cidades, um exemplo disso é como a Amazon deseja fazer entrega de mercadoria no futuro, através da **Amazon prime Air**[37]. Os drones vão começar a fazer entregas de forma mais rápida do que alguma vez se pensou fazer. Os drones também vão entrar noutras áreas como na saúde, como no exemplo do drone ambulância [38] ou drones capazes de resgatar pessoas, ou até drones veículos[39].

Como foi visto anteriormente, um conceito importante das **SC** é ***smart people***, ou seja, o conhecimento vai ser um fator fundamental no melhor funcionamento da cidade, tanto a nível dos cidadãos como do funcionamento da cidade (exemplo: saber onde ocorre os desperdícios na cidade). Assim tudo o que sejam tecnologias ou iniciativas que contribuam para que o cidadão obtenha conhecimento, ou permita com que o cidadão possa participar de uma forma mais ativa no desenvolvimento da cidade, tudo contribui para uma cidade mais inteligente. Exemplo de tecnologias que ajudam nesta área, é a existência de inúmeros cursos e aulas online, que existe na internet, desde **Kahn Academy**, **MIT OpenCourseWare**, **Coursera** até **StackOverFlow** que ajuda em qualquer questão na área de programação entre outros. Não é novidade que a internet é hoje em dia uma grande fonte de conhecimento, mas ser desenvolvedor não fica só por saber. Desde impressoras 3D a dispositivos como **Arduino** ou **Raspberrypie**, qualquer um com um pouco de treino e principalmente vontade, consegue fazer aplicações ou dispositivos IoT que ajudarão as cidades a serem mais eficientes. Concluindo, com plataformas como **Kickstarted** ou **Patreon**, entre outras de *CrowdFunding*, qualquer pessoa ou equipa pode obter investimento para realizar projetos que beneficiarão a cidade.

Assim conclui-se a descrição de algumas tecnologias e iniciativas inovadoras das **SC**.

2.3 JOGOS SÉRIOS SOBRE SMART CITIES

O que se pretende fazer é um jogo que seja divertido é ao mesmo tempo tenha uma componente pedagógica em relação às **SC**. Que jogos já foram feitos com a mesma finalidade ou parecida? Nesta secção os jogos serão avaliados tendo em conta os seguintes critérios, aspeto gráfico, jogabilidade/interação, aspeto educacional/pedagógico.

2.3.1 Smart Cities Ready To Go!

Figura 2- Aplicação Smart Cities, Ready to Go!

A Figura 2 é uma aplicação para o telemóvel de educação e de realidade aumentada. O objetivo da aplicação e a demonstração de soluções inteligentes de energia de modo a reduzir o nível de CO2 nas cidades. O objetivo do jogo é reduzir o número de emissões de CO2 instalando medidas energéticas eficientes na cidade[40]. A aplicação é mais educacional do que um jogo, visto que tem um baixo nível de desafio e interação (jogabilidade), mas explica de uma forma lúdica muitos conceitos energéticos sobre as **SC**.

Design/aspeto gráfico: tem um aspeto simplista, com pouco detalhe, mas suficiente para perceber tudo. Apesar de os modelos 3D serem simples e sem grandes texturas, tudo é agradável à vista. A interface utilizador está muito bem-feita, sendo bastante intuitiva e simples, de modo que qualquer utilizador consegue usar sem grandes dificuldades ou tutoriais.

Jogabilidade/Interação: em termos de interação e jogabilidade é bastante reduzido, a tarefa do jogo é reduzir emissões de CO2 instalando novas tecnologias renováveis usadas nas **SC**. Para instalar essas tecnologias, basta arrastar do menu para o modelo 3D do edifício e conclui-se a instalação. Em termos de interação, o jogador também pode rodar os modelos 3D e fazer aproximações de modo a poder visualizar melhor os modelos. O jogo também inclui uma versão de realidade aumentada, que permite ao jogador ver os modelos 3D no mundo real através da câmara do telemóvel ou tablet.

Aspeto pedagógico/educacional: neste setor o jogo comporta-se muito bem, explicando como todas as medidas realizadas no jogo ajudam a reduzir o CO2 nas cidades. Também ensina tecnologias usadas hoje em dia para produzir energia renovável. A questão mais interessante, a nível pedagógico, é que todas as medidas que o jogador tem de implementar, são casos reais já implementados em cidades no mundo e tudo é muito bem explicado na aplicação, tendo até hiperligações para vídeos que explicam melhor algumas das iniciativas.

Impressões finais: Este jogo é mais aplicação educacional do que jogo, pois não existe desafio. Em termos pedagógicos é muito bom, mas em termos de jogo falta-lhe muito. Assim sendo, o jogo desenvolvido nesta dissertação, pretende ter mais interação comparativamente a esta aplicação.

2.3.2 SMART CITY SERIOUS GAME

Figura 3- Jogo sério sobre Smart Cities em html

Este é um jogo sério sobre smart cities que se pode encontrar neste site[41]. O objetivo do jogo é resolver problemas na cidade usando a tecnologia adequada para resolver o problema. Basicamente é tornar uma cidade "normal", numa inteligente, de modo a fazer face aos problemas que a cidade tem. O jogo baseia-se como no anterior numa mecânica "*Drag and Drop*", ou seja, arrastar as tecnologias de um menu para resolver problemas concretos. No fim de cada nível, será dada uma pontuação dependendo das escolhas feitas, e o jogador terá de realizar um questionário (quiz), ao qual tem de acertar para passar de nível. Este já é mais parecido com um jogo sério, e testa realmente se aprendemos alguma coisa ou não, mas é apenas um demo e não um jogo completo.

Design/aspeto gráfico: em termos gráficos, é muito simples e rudimentar, usa uma imagem para representação da cidade e menus como estão representados na Figura 3.

Jogabilidade/Interação: tal como no jogo/aplicação anterior, baseia-se essencialmente em arrastar tecnologias para o mapa da cidade. Neste jogo, esta mecânica está melhor resolvida já que inclui um desafio, ou seja, para cada problema da cidade, tem-se de usar a tecnologia certa, entre várias opções. No final de cada nível como já foi referido, aparece um quiz final testando conhecimentos obtidos. O jogo, por ser mais um demo do que um jogo completo, só tem dois níveis, o que o torna muito curto.

Aspeto pedagógico/educacional: Este jogo como o anterior são bastante bons, nesta área. Este incide sobre muito mais tecnologias de **SC** do que o jogo anterior, desde IoT, serviços cloud, segurança informática, telecomunicações, energia, etc… falando sempre de tecnologias também já existentes. Explica qual o propósito de cada um deles e que problema pretende resolver.

Impressões finais: Este é um verdadeiro jogo sério sobre **SC**, só tem como problema a falta de interação e jogabilidade, que torne o jogo mais apelativo. Também é demasiado curto, o que torna o jogo ainda menos cativante. Nesta dissertação, pretende-se realizar um jogo como o

mesmo nível pedagógico do que este, mas mais apelativo e longo, por exemplo sendo um jogo de final aberto.

2.3.3 Watch Dogs 2014/2016

Figura 4- Imagem do jogo Watch Dogs

Este Jogo já não é tão educacional como os anteriores, mas não deixa de explorar bem os conceitos das **SC**. O jogo desenvolve-se há volta de uma personagem que é um hacker, numa cidade inteligente. Sendo assim ele consegue manipular quase todos os objetos da cidade, que são controlados por um único sistema operativo que controla a cidade inteira. É um franchise criado pela Ubisoft (empresa de jogos) que investiu bastante dinheiro na criação deste franchise (que já conta com dois jogos), por isso também tem bastante qualidade em termos técnicos.

Design/aspeto gráfico: sendo um jogo feito com alto investimento, em termos gráficos é muito bom, desde a iluminação, aos modelos 3D de uma cidade inteira, tem tudo imenso detalhe. A cidade parece viva com pessoas e carros a andar de um lado para o outro na cidade. Os controlos são intuitivos e simples.

Jogabilidade/Interação: em termos de Jogabilidade, o jogador tem controlo de uma personagem que para se deslocar na cidade, pode usar *parkour* e conduzir veículos terrestres ou aquáticos. Como ênfase principal, o jogo foca-se no combate furtivo e disparo de armas de fogo, tendo também uma componente online que permite jogar com outros jogadores no mundo. O elemento diferenciador deste jogo em relação aos outros jogos, é o facto de o jogador poder utilizar a cidade a seu favor, de modo a completar os seus objetivos (exemplo: manipular semáforos da cidade para provocar um acidente a um criminoso).

Aspeto pedagógico/educacional: este jogo foca-se mais na componente de entretenimento do que do conhecimento, visto que é um jogo feito com esse propósito. Apesar disso, consegue demonstrar muito bem como uma cidade inteligente funciona, desde interligação e comunicação de diversos dispositivos eletrónicos, ao uso de smartphone para navegar melhor na cidade, até há segurança informática o jogo consegue de uma forma indireta, com que o jogador aprenda muitas coisas relacionadas com **SC** jogando. Tem problemas, pois muitas vezes, não representa bem a realidade, esta é a sua maior falha na parte pedagógica, mas compreensível pois muitas das vezes que isso acontece é para tornar o jogo mais fluido ou mais lúdico.

Impressões finais: Este jogo tem um fator muito importante que se pretende copiar nesta dissertação que é ensinar quase por "osmose", ou seja o jogador no meio do entretenimento está a aprender. Watch Dogs consegue fazer isso, muito bem, mas para ser um jogo sério teria de dar muito mais foco à parte pedagógica do que a do entretenimento, o que não faz. Esta dissertação pretende ter uma componente pedagógica mais forte do que no Watch Dogs, mas também deverá ser divertido e cativante de jogar.

2.3.4 IBM City One (2010)

Figura 5 - Jogo sério da IBM

Este é um jogo sério criado pela IBM, onde o jogador faz o papel de presidente de uma cidade, portanto tem como objetivo gerir e melhorar a cidade. O jogo realiza-se num número limitado de turnos, onde o jogador tem de tomar decisões para resolver problemas da cidade. O jogador tem ao seu dispor um conjunto de consultores nas áreas de energia, água, retalho e bancário, eles vão propor soluções e cabe ao jogador tomar a melhor opção de modo a melhorar a cidade. Este jogo lida com vários conceitos sobre as **SC**, desde energias renováveis, coleção de dados da cidade, IoT, entre outros. Este jogo tem como elemento diferenciador, tentar ser o mais realístico possível nos problemas e decisões a tomar para a cidade. Concluindo, este é um bom jogo educacional, tendo em conta que o grande foco da jogabilidade tem a haver com a parte pedagógica, mas em termos visuais, não é o mais apelativo apesar de não ser mau.

Design/aspeto gráfico: graficamente o jogo é muito simples sendo uma imagem de uma cidade, o jogador pode navegar pela cidade usando o cursor e a banda sonora é agradável. Em relação aos menus do jogo, são intuitivos e simples.

Jogabilidade/Interação: em relação à interação do jogador é muito básica, é simplesmente escolher as melhores opções que os consultores da cidade sugerem e ver se conseguimos melhorar o nível de vida na cidade. Através de gráficos que o jogo disponibiliza, é possível ver se a cidade está a melhorar ou não. O jogo traz também um componente social/online, pois pode-se ver num ranking global como nos saímos no jogo em relação a outros jogadores.

Aspeto pedagógico/educacional: o foco do jogo é no planeamento de cidades, mas usa muitas soluções inteligentes e realistas usadas nas **SC**, para resolver os diversos problemas que o jogo apresenta das cidades. Tornando assim o **IBM City One** um verdadeiro jogo sério.

Impressões finais: Este jogo assemelha-se ao que se pretende realizar nesta dissertação, sendo que a sua componente pedagógica um ponto bastante forte. Esta dissertação pretende conter conteúdo realístico sobre as **SC**, como este jogo, mas adaptá-lo mais para todas as idades

(especialmente crianças/adolescentes), fazendo com que seja mais interativo e tenha informação mais simples.

2.3.5 Block'hood

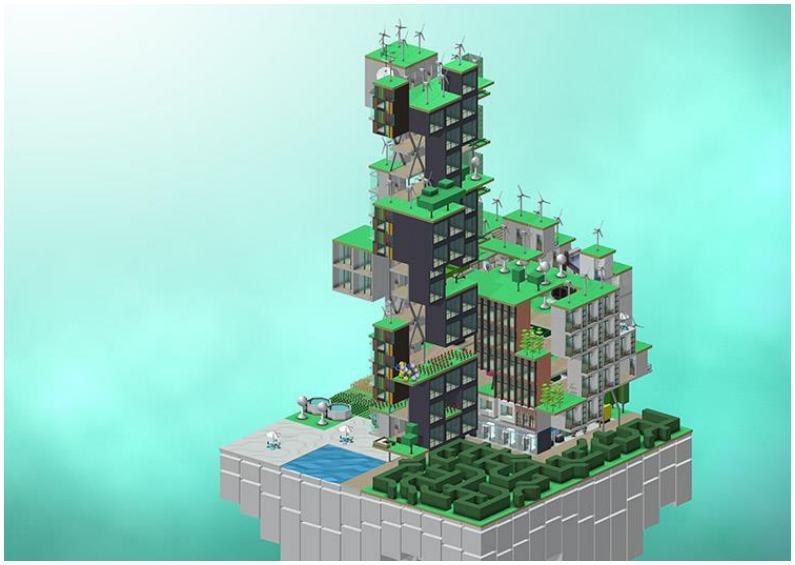


Figura 6 - Um bairro do jogo Block'Hood[42]

Este jogo é um simulador de construção de um bairro em blocos. O objetivo do jogo é criar um bairro em blocos que seja sustentável e crie abundância. "Cada Bloco que se pode criar tem Entradas e Saídas. Por exemplo, uma árvore pode precisar de água para criar oxigênio, e uma loja pode precisar de consumidores para criar dinheiro. Ao compreender como cada bloco é dependente de outros blocos, o jogador pode criar uma rede produtiva" [42]. "Se os blocos não obtiverem as entradas de que necessitam, eles vão decair ao longo do tempo, lentamente tornando-se abandonados ou destruídos" [42].

Design/aspeto gráfico: Como se pode ver na Figura 6 o jogo é bastante apelativo à vista, mesmo tendo um design muito simples. Em termos de interface utilizador – jogo, é muito simples sendo composto por um menu com os blocos existentes, onde basta arrastá-los para o terreno do bairro para os usar, e um menu que mostrar os recursos que o bairro produz e necessita para sobreviver.

Jogabilidade/Interação: em termos de interação do jogador, este consegue criar o seu bairro com absoluta liberdade na utilização dos blocos que quer, tendo à sua disposição mais de 96 blocos únicos para utilizar [42]. Tem 3 modos de jogo um educacional, outro livre e um de desafios[42].

Aspeto pedagógico/educacional: como foi referido anteriormente, este jogo tem um modo educacional que consiste no seguinte: "Modo de educação é um tutorial estendido que leva o jogador em uma jornada, olhando tecnologias atuais e como eles funcionam" [42]. Só por si o jogo já tem bastantes fatores de aprendizagem sobre **SC**, visto a importância que dá ao uso energias renováveis e ao equilíbrio que existe entre a cidade e a natureza.

Impressões finais: este jogo tem características muito semelhantes ao que esta dissertação pretende realizar, misturando muito bem a componente de entretenimento com a de educação. Esta dissertação pretende se diferenciar deste jogo, dando mais importâncias aos elementos de **SC** do que neste jogo.

2.3.6 Simuladores De Criação De Cidades

Figura 7 - Imagem do jogo SimCity 2013[43]

Jogos de simulação de cidades existem muitos, por isso este capítulo pretende analisar de um modo geral este tipo de jogos principalmente os mais conhecidos como **SimCity** e **City Skylines**. Neste tipo de jogo, o jogador costuma fazer o papel de presidente da cidade, com o objetivo de criar a melhor cidade possível.

Design/aspeto gráfico: se considerarmos os jogos mais conhecidos referidos anteriormente, visto que são jogos de alto investimento, o aspeto gráfico dos jogos é muito bom, tendo diversos modelos 3D de edifícios, bem detalhados. Em termos de interface utilizador, apesar de os jogos terem algum nível de complexidade e, portanto, muitos menus, estão cada vez mais fáceis de usar, sendo contextual o aparecimento de menus[44].

Jogabilidade/Interação: a liberdade de criação destes jogos é imensa, desde desenhar estradas, criar edifícios, moldar o terreno etc. A simulação da cidade é a componente principal do jogo, sendo que varia de jogo para jogo a sua qualidade. Dentro da simulação, jogos como SimCity e City Skylines, simulam desde tráfego, economia, energia, segurança, poluição, saúde, etc.

Aspeto pedagógico/educacional: mais uma vez, o jogo mistura muito bem os elementos educacionais e de entretenimento na perfeição, sendo muitas vezes um dos primeiros contactos dos jovens para saber como funciona uma cidade. Como foi referido anteriormente o jogador para vencer no jogo tem de saber como planear a cidade, e ter preocupações como impostos, dinheiro, educação, energia, saúde, segurança, entre outros.

Impressões finais: este tipo de jogo é muito semelhante ao que se pretende fazer nesta dissertação, principalmente na parte de educar de forma divertida. Como elemento de diferenciação de estes jogos, esta dissertação tem como propósito dar menos ênfase à parte de planeamento de cidades (onde estes jogos têm o seu maior foco) e mais nos conceitos sobre as **SC**.

Finalizando o capítulo, esta dissertação tenciona realizar um jogo na mesma linha que o Sim City e comentários descritos nas **impressões finais** anteriores, mas focado nos aspetos das cidades inteligentes.

Tabela 1 - Avaliação dos jogos encontrados sobre o tema das SC

Jogos	Avaliação (Bom, Médio, Fraco)		
	Design/aspeto gráfico	Jogabilidade/Interação	Aspeto pedagógico/educacional
Smart Cities Ready to Go!	Bom	Fraco	Bom
Smart City Serious Game	Fraco	Médio	Bom
Watch Dogs	Bom	Bom	Médio
IBM City One	Médio	Bom	Bom
Block'Hood	Bom	Bom	Bom
Sim City e City Skylines	Bom	Bom	Bom

2.4 Ferramenta de Criação de Jogos ou "Game Engine"

Para procurar um programa, que facilite a criação de jogos, preferiu-se encontrar uma ferramenta que seja gratuita, tenha uma ampla comunidade de utilizadores e tutoriais, conseguisse modelar jogos 2D como em 3D e que exporte os jogos para o maior número de plataformas possível. Tendo estas condições iniciais de procura, encontrou-se as seguintes ferramentas: "Unity", "Unreal Engine 4", "Blender". Todas estas ferramentas, são gratuitas, apesar de terem versões profissionais pagas, e têm grandes comunidades de utilizadores com tutoriais. Tendo experimentado cada uma das ferramentas, serão apresentadas as qualidades e defeitos de cada uma delas, sendo que no fim será escolhida uma.

2.4.1 Blender

Blender é mais usado como uma ferramenta de modelação 3D e de animação, do que usado como "***Game Engine***", apesar de possibilitar isso. Como qualidades, o Blender possibilita desde o desenho de objetos 3D/2D, animações, simulação de física e de partículas, também tem uma grande comunidade de utilizadores e vários tutoriais na internet. Como aspetos negativos, o Blender tem uma interface utilizador muito difícil de usar. Não consegue exportar os jogos para muitas plataformas, o que torna o Blender mais uma ferramenta de criação de modelos e animação 3D/2D do que propriamente uma ***Game Engine***.

2.4.2 Unreal Engine 4

Unreal Engine 4 é uma ***Game Engine*** muito poderosa, sendo uma das ferramentas de criação de jogos preferida de grandes empresas. Tal como o Blender consegue criar objetos 3D/2D e animá-los, apesar de neste aspeto ser pior que o Blender, mas é mais fácil de usar, consegue simular física e partículas, e exporta para os diversos dispositivos usados para jogos. Em contrapartida, ainda é muito difícil de usar, sendo mais usado por empresas criadoras de jogos.

2.4.3 Unity

Unity apesar de ter pior prestação a nível gráfico e de processamento em relação ao Unreal Engine 4, o Unity é claramente a ferramenta mais fácil de usar, tendo inúmeros tutoriais na internet e uma loja com diversos exemplos de jogos feitos. Também consegue modelar objetos 3D/2D e animá-los, mais uma vez é pior que o Blender nisto, também consegue simular física e partículas. Por fim consegue exportar para todas as plataformas que possibilitem jogos. Sendo o Unity uma *game engine* feita para a rápida prototipagem, será usada esta ferramenta como a principal para criação do Jogo.

Tabela 2 - Comparação de Game Engines

Game Engine	Avaliação					
	Custo	Curva de Aprendizagem	Processamento e Gráficos	Simulação de Física e partículas	Modelação e animação	Exportação dos jogos
Blender	Grátis	Difícil	Bom	Bom	Bom	Para computador
Unreal Engine 4	Grátis	Média	Bom	Bom	Médio	Para todos os dispositivos
Unity	Grátis	Fácil	Bom	Bom	Médio	Para todos os dispositivos

3 SMART CITY BUILDING GAME

3.1 INTRODUÇÃO

Tendo analisado diversos jogos no estado de arte, pôde-se retirar ideias para o jogo a realizar. Em primeiro lugar deve-se estabelecer objetivos que o jogo deve cumprir, os objetivos são os seguintes:

1. Coexistência tanto de uma componente séria como lúdica no jogo.
2. Tema central do jogo são as Cidades Inteligentes.
3. Não pode ser muito complexo, tanto na parte educativa como na jogabilidade, de modo a ser acessível para pessoas de todas as idades, especialmente jovens/crianças.
4. Tem de ser acessível de instalar e jogar ("plug and play"), quer seja para dispositivos móveis ou computadores, tem de ser fácil de instalar e jogar.

Estes são os requisitos mais gerais a ter em conta na realização do jogo. Um requisito menos importante é em que língua se realizará o jogo, foi decidido que Inglês era uma boa opção visto que abre a possibilidade de qualquer pessoa no mundo puder jogar o jogo.

A ideia que surgiu para realizar o jogo, veio da junção de dois jogos analisados no estado de arte, o BlockHood e os jogos de simulação de Cidades, especialmente o mais conhecido Sim City.

De Blockhood retirou-se a mecânica central do jogo, que é que, para construir um bairro é necessário criar diversos blocos, que estão todos dependentes uns dos outros para sobreviver. Cada bloco consome e produz algo, o trabalho principal do jogador é gerir os blocos de modo a criar um bairro sustentável. Usar esta mecânica é o que se pretende fazer no jogo a realizar sobre Cidades Inteligentes. Será explicado com mais detalhe, mais à frente neste capítulo.

De Sim City retirou-se a mecânica de construção e gestão de cidades, e também o conceito de jogo com final aberto. Este conceito de final aberto pode ser resumido no seguinte: é um jogo onde se pode perder, mas não se pode ganhar, ou ganhar é um conceito relativo ao jogador. No caso do jogo do Sim City, é possível perder o jogo se deixarmos a cidade ir à bancarrota, por outro lado o jogo não tem fim, ou seja, só acaba quando o jogador achar que criou a cidade que desejava. Obviamente existe sempre um objetivo, que neste caso é criar uma cidade sustentável, ou seja que gere receita suficiente para pagar as suas despesas. Este pode ser um final possível, mas o jogador é que toma a decisão final se está satisfeito com a sua cidade ou deseja construir uma ainda melhor e maior. Este conceito de final aberto será adaptado no jogo a realizar.

3.2 ADAPTAÇÃO E CONCEITO

Com base no que foi previamente anunciado, aqui descreve-se o conceito principal do jogo. Em primeiro lugar o objetivo do jogo será o de criar uma cidade e geri-la, de modo a que seja sustentável. Para simplificar a implementação e a jogabilidade deste jogo, será adaptada a mecânica que BlockHood usa, onde cada bloco consome e produz algo. No diagrama em baixo será esclarecido como será adaptado este conceito.

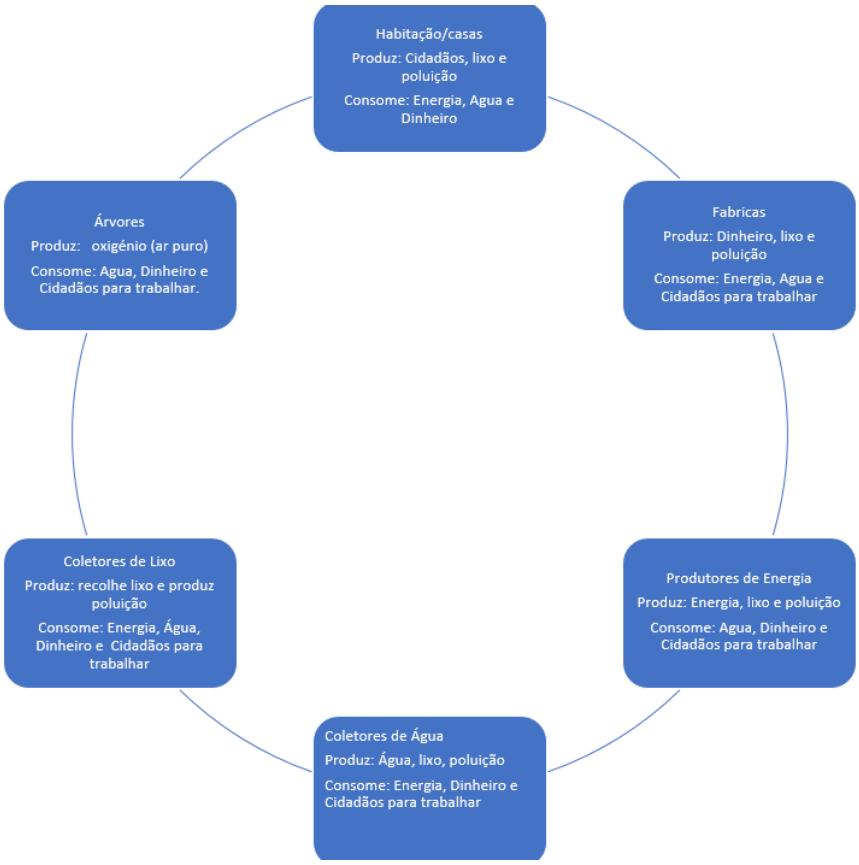


Figura 8 - Diagrama da mecânica do jogo

Como se pode ver na Figura 8, cada um dos componentes produz algo e consome algo e todos precisam uns dos outros para a cidade progredir. Por exemplo, Habitações produzem cidadãos, mas para produzir habitantes para a cidade, é necessário dinheiro, água, energia, ar puro, etc. Ou seja, para construir uma casa também é necessário fábricas que produzam dinheiro e fontes de energia. Este diagrama, não será a versão final desta mecânica (apresentada na Figura 8), mas representa bem o conceito que se pretende implementar no jogo. Assim, o jogador terá o desafio principal de criar uma cidade usando estes componentes de modo a que seja sustentável.

Em todos os jogos é necessário existir uma forma de ganhar ou perder o jogo. Como se pretende realizar um jogo de final aberto, é necessário dar mais importância a como se perde, do que como se ganha. No SIM City é possível perder, se a cidade se endividar muito. No jogo que se pretende realizar, quer-se dar mais importância a outros fatores sem ser só o dinheiro. No estado de arte foi referido que o objetivo principal das cidades inteligentes é no fundo melhorar o nível de vida dos seus cidadãos, ou seja torná-los mais felizes. A felicidade apesar de ser um conceito abstrato, é algo que todas as pessoas conseguem entender de algum modo, por isso decidiu-se criar um indicador de felicidade no jogo. Este indicador estará diretamente relacionado ao modo como o jogador gere a cidade, se gerir bem, por exemplo a cidade não tem desemprego, produz dinheiro suficiente para os seus gastos, etc… este indicador é positivo, caso contrário, se gerir mal, que é exatamente o oposto do descrito previamente, então o indicador é negativo. Se se perdesse o jogo, sempre que este indicador fosse negativo, tornaria o jogo demasiado difícil para se jogar, portanto, decidiu-se criar outro indicador designado de "Saúde da Cidade". Este indicador, será uma barra por exemplo de 0 a 100, mas pode abranger valores diferentes, e é reduzido sempre que o indicador "Felicidade da Cidade" for negativo e aumenta (até ao seu limite), caso seja positivo. Assim obtém-se uma forma razoável de o jogador perder o jogo e de o "vencer". Por exemplo, se o indicador "Saúde da Cidade" for igual ou inferior a zero então perde-se o jogo, caso contrário permanece-se nele. Assim está resolvido o problema de ganhar ou perder o jogo, na parte de ganhar como é de final aberto, digamos que o objetivo principal é criar uma cidade sustentável, mas se o jogador quiser, criar uma cidade maior e melhor deve poder fazê-lo. Para marcar um ritmo ao jogo, o jogador terá de jogar por turnos limitados no tempo, como também acontece, mais ou menos, nos jogos de simulação de cidades.

3.3 PARTE EDUCACIONAL

Em relação à parte educacional, decidiu-se focar em alguns conceitos fundamentais sobre as **SC**, tais como:

1. O problema que levou à criação do conceito das **SC**.
2. Ensinar como a obtenção de informação (Big Data, IoT), ou melhor conhecimento, sobre a cidade, através do uso de IoT, pode ajudar a fazer uma gestão mais inteligentes dos recursos da cidade.
3. Explicar a importância do meio ambiente e da educação nas **SC**.

Estes são os conceitos que o jogo pretende ensinar com maior intensidade, mas poderão ser mais ou menos na implementação final do jogo. Como ensinar estes conceitos? Em seguida será exposto, qual a proposta que se pretende usar para explicar estes conceitos.

3.3.1 Como explicar o problema que levou à criação do conceito das SC?

Sendo difícil ensinar esta questão de uma forma explícita no jogo, optou-se então, por realizar um pequeno vídeo introdutório no jogo (como aliás é normal fazer na maior parte dos jogos), não muito mais que um minuto, só para explicar brevemente esta questão aos jogadores.

3.3.2 Como ensinar sobre a importância que tem a recolha de informação sobre a cidade, no funcionamento de uma SC?

Neste caso, encontrou-se uma maneira de ensinar esta questão, usando uma mecânica de jogo. A ideia é a seguinte, neste tipo de jogos de simulação de cidades, o jogador tem de saber se a cidade está a funcionar bem ou mal, para isso normalmente existem dados que indicam isso mesmo, por exemplo, se existe água ou a sua falta, se existe energia ou a sua falta, etc. O que se pretende fazer de diferente dos restantes jogos, é explicar de onde vêm esses dados. Por exemplo, no menu onde o jogador pode comprar uma habitação ou um edifício, pode-se explicar com um pequeno texto como a habitação inteligente envia informação, sobre o seu funcionamento, a uma base de dados que a cidade gere. Daqui surgiu uma nova ideia a implementar no jogo. Usando a referência [6], a IBM tem uma solução para as cidades inteligentes, que passa por usar um sistema central de gestão da cidade, que no fundo é um edifício que recolhe e gere informação sobre a cidade, de modo a poder tomar melhor decisões, baseados nos dados recolhidos. Portanto decidiu-se implementar este sistema de gestão central no jogo.

Como implementar este sistema de gestão da cidade, no jogo? A solução encontrada é a seguinte: este sistema também será um dos blocos (habitação, fábricas, etc.) que o jogador poderá comprar. Será apresentado como um edifício, mas terá um comportamento diferente dos outros componentes da cidade, apresentados na Figura 8. Por exemplo, poderá ou não consumir e produzir coisas, terá de ser o primeiro componente a ser construído antes de construir o resto da cidade, e por fim todos os outros componentes da cidade (edifícios, fábricas, painéis solares, etc.) terão de estar ligados, a ele, de alguma forma (estradas, fios elétricos, etc.), de modo a que seja óbvio que este edifício está ligado aos outros, portanto pode existir comunicação entre eles.

Assim é possível resolver dois problemas ao mesmo tempo. O primeiro é o de ensinar este conceito de recolha de informação, o outro é de obrigar o jogador a criar uma cidade que faça sentido. O que se considera como uma cidade que faça sentido? É uma onde todos os edifícios estão conectados uns aos outros, pelo menos por uma estrada. Como todos os edifícios, painéis solares, repositórios de água, etc… estarão ligados ao edifício gestor central da cidade (através de estradas), isto obriga a que todos os edifícios estejam ligados contiguamente por estradas uns aos outros. Por exemplo, caso o jogador esqueça-se de ligar um edifício por estradas ao gestor da cidade, então deixa de estar ativo na cidade (deixa de produzir ou consumir recursos para a cidade).

3.3.3 Como explicar a importância do meio ambiente e da educação nas SC?

Em relação ao meio ambiente, isso é explicado graças ao funcionamento da mecânica principal do jogo, ou seja, cada componente da cidade gera poluição e lixo, só se o jogador se preocupar com estes elementos é que pode vencer no jogo, caso contrário, o indicador “Felicidade da Cidade” será negativo e por sua vez a “Saúde da Cidade” diminuirá. Isso obriga o jogador a ter uma preocupação razoável com o meio ambiente. Na questão da educação, a solução encontrada é a seguinte: cada componente da cidade quer seja habitação, fábricas, painéis solares, entre outros, terão por exemplo três versões, ou níveis. O primeiro nível sendo a pior versão desse componente, por exemplo o primeiro nível da fábrica é muito poluente e produz pouco dinheiro, enquanto que o terceiro nível já produz muito dinheiro e consome poucos recursos. Para comprar uma fábrica de nível 3, o jogador precisa de ter pontos de educação, que se podem obter através da criação de escolas ou centros de investigação, que

seriam mais um dos blocos que o jogador pode comprar. Assim o jogador tem um bom pretexto para preocupar-se com a educação, pois só através destes pontos é que pode comprar melhores versões de cada edifício.

3.4 ACESSIBILIDADE

Neste subcapítulo, pretende-se discutir qual a plataforma eletrónica ideal para exportar o jogo, desde dispositivos móveis a computadores. O que se pretende fazer é um jogo que dê para o maior número de dispositivos eletrónicos possíveis. A ferramenta de criação de jogos escolhida "Unity" consegue exportar jogos para todas as plataformas, mas é necessário fazer sempre ajustes, muitas vezes até difíceis, para exportar por exemplo para dispositivos móveis ou Computadores. Tendo em conta esta dificuldade tentar-se-á, por exemplo, fazer a interação do utilizador-jogo, o mais táctil possível, ou seja, fazer o jogo de modo a que o jogador só tenha de usar, o rato ou os dedos, para jogar, em vez de necessitar de comandos ou teclado. Pretende-se dar mais importância à realização do jogo, de forma a ser jogável em plataformas móveis (telemóveis, tablets) pois é onde o mercado de jogos se encontra mais forte hoje em dia.

3.5 PROPOSTA DE DESENVOLVIMENTO

Neste subcapítulo, deseja-se expor uma proposta de desenvolvimento do jogo descrito nos subcapítulos 3.2 e 3.3. Para isso pretende-se desenvolver os seguintes temas:

- Interface entre jogador e o jogo
- Classes e estruturas usadas para desenvolver o jogo descrito
- Fluxograma do funcionamento geral do jogo.

3.5.1 Interface

Propõe-se realizar os seguintes menus, para o jogador interagir.

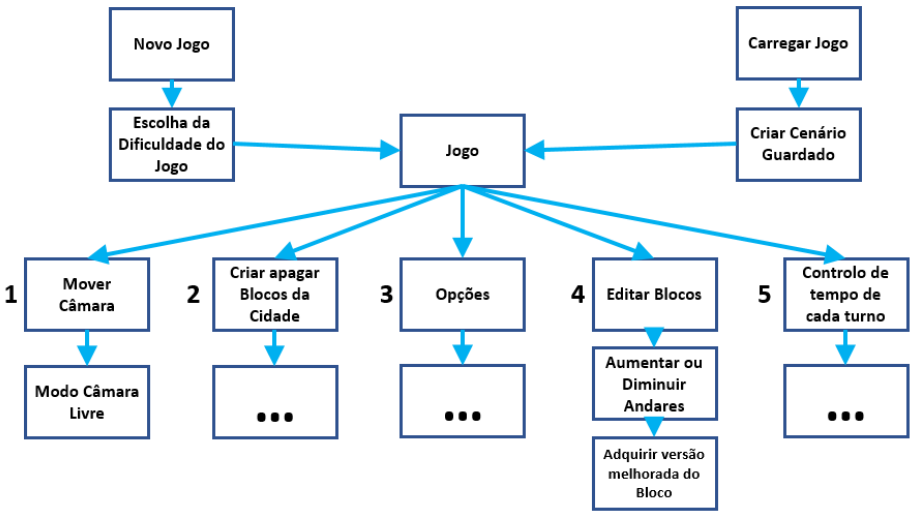


Figura 9 -Menus de Interface Jogador-Jogo

Na Figura 9, pode-se ver os menus que o jogador terá acesso para poder jogar. No início o jogador poderá escolher duas coisas, ou inicia um novo jogo, ou carregar um jogo previamente guardado. Se começar um novo jogo, então terá de escolher o seu nível de dificuldade. A dificuldade do jogo vai se basear na **quantidade de dinheiro** e de "**Saúde da Cidade**" com que se inicia o jogo, também irá afetar os custos educacionais para obter versões melhoradas de cada edifício. Durante o jogo, o jogador deverá poder manipular a câmara do jogo, para visualizar a cidade construída, e poder criar ou apagar edifícios, estradas e componentes da cidade. Os menus 1,2 e 4 da Figura 9 vão permitir isso.

No menu 2 o jogador poderá escolher que edifício criar na cidade bastando clicar na imagem do objeto a construir e de seguida, clicar no mapa do jogo. Para editar um bloco, o jogador deverá aceder ao menu 4 e depois clicar num bloco do mapa, depois deverá escolher um dos submenus do menu 4.

O menu 5 serve para o jogador poder controlar, o tempo que demora cada turno, podendo mesmo escolher, que o tempo deixe de passar completamente. O controlo do tempo é importante, como já foi descrito anteriormente, pois a cidade só progride se os turnos forem passando. O menu das opções servirá para colocar todas as interações do jogador que não estejam diretamente relacionadas com a construção da cidade, como por exemplo guardar a cidade construída até ao momento, desligar ou ligar o som, sair do jogo, etc…

3.5.2 Classes & Estruturas De Dados Importantes

Neste subcapítulo pretende-se descrever algumas classes e estruturas de dados que permitam desenvolver o jogo proposto. Existirá uma classe principal responsável por correr todo o código usado para jogar o jogo, podemos chamar-lhe de "**Classe Principal**" ou "**Main Class**". Em seguida será descrito as classes principais que serão usadas na "Classe Principal".

Matriz "Cidade": matriz do tipo object, que em C#, é uma matriz que aceita receber todo o tipo de dados. Esta matriz vai conter toda a cidade criada pelo jogador. O mapa do jogo será um quadriculado onde cada posição pode conter um objeto/bloco da cidade.

Figura 10 - Exemplo do mapa do jogo

Matriz de "Identificação da Cidade": matriz do tipo inteiro que identifica o que se encontra em cada posição da matriz "Cidade". Por exemplo um bloco estrada é identificado pelo número 2 e o terreno é identificado pelo número 0.

Classe Interação do jogador: classe que controla o input que o jogador dá com o cursor e mapeia para ações e posições no mapa do jogo.

Variáveis:

- **Câmara do jogo**
- **Cubo Vermelho:** uma das variáveis que esta classe tem, é um cubo vermelho, que tem como propósito, indicar ao jogador a posição do mapa que o jogador clicou com o cursor.

Métodos: esta classe tem apenas um método que corre uma vez por cada ***frame*** (imagem produzida pelo jogo). Este método tem o propósito de calcular a posição do mapa onde o jogador clicou com o cursor.

Classe "Bloco Jogável": classe que contém toda a informação sobre os objetos que o jogador pode construir. À medida que o jogador vai criando edifícios na cidade, vai construindo instâncias desta classe e guardando na matriz "**Cidade**", por fim vai atualizar a matriz de "**Identificação da Cidade**".

Variáveis:

- **Tipo de bloco:** variável do tipo inteiro que serve para identificar o bloco/objeto que o jogador pode criar, que pode ser uma habitação, uma fábrica, um painel solar, etc…
- **Conectado:** variável do tipo booleano que indica se o bloco/objeto está conectado ao edifício gestor da cidade por estradas.
- **X e Z:** duas variáveis do tipo inteiro responsáveis por identificar a posição do objeto na matriz "Cidade" e consequentemente a sua posição no mapa do jogo.
- **Custo:** variável do tipo inteiro que indica o custo de compra do objeto/bloco.
- **Nome:** variável do tipo *string* que guarda o nome do bloco/objeto
- **Altura:** variável do tipo inteiro que indica o número de andares do edifício.
- **Vetor de Consumo e Produção:** é um vetor do tipo inteiro que guarda o que o objeto consome e produz da seguinte forma: cada posição do vetor guarda um inteiro relacionado com um recurso que o jogador tem de se preocupar. Por exemplo a posição 0 do vetor indica o consumo energético do bloco/objeto. Caso o valor dessa posição seja negativo, então consome energia, caso contrário produz. O mesmo acontece para as outras posições do vetor, mas com recursos diferentes.

Métodos:

- **Construtor:** recebe como parâmetros de entrada o tipo de objeto, o nome a posição e o custo, inicializando essas variáveis.
- **Está conectado:** função que recebe as duas matrizes principais previamente descritas e altera a variável booleana que indica se o bloco/objeto está conectado.

Classes Descendentes da classe "Bloco Jogável": Todos os objetos/blocos que o jogador pode criar são descendentes da classe "Bloco Jogável" pois esta classe contém toda a informação sobre o objeto.

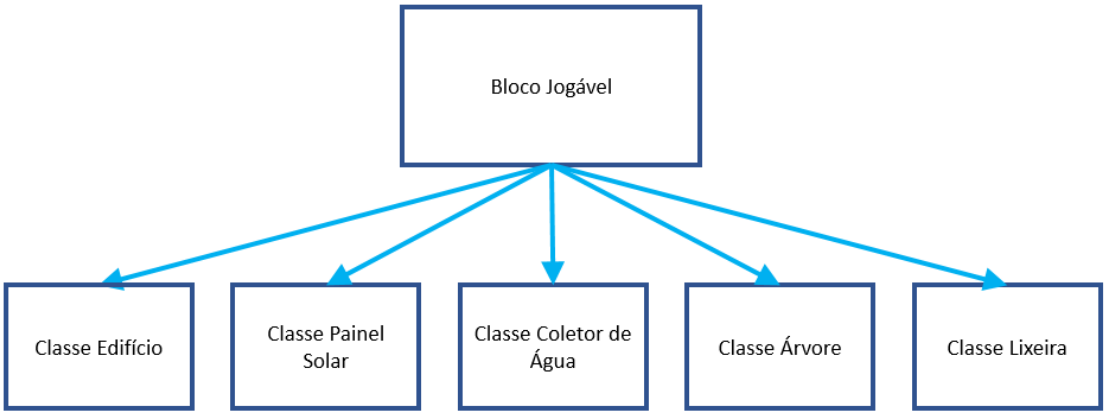


Figura 11 - Descendência da classe Bloco Jogável

Pretende-se fazer assim o jogo, de modo a que sempre que o jogador crie algum objeto, por exemplo um painel solar, então a classe painel solar é responsável por saber criar o objeto (forma geométrica) e apagá-lo, e por ser descendente da classe "Bloco Jogável" então contêm também toda a informação útil sobre esse objeto.

Cada uma destas classes "filho" terá a seguinte estrutura geral:

Variáveis: para além de todas as variáveis da classe "pai", terão variáveis para guardar a estrutura ou a forma geométrica do objeto que o jogador irá visualizar. Por exemplo a classe Edifício terá de guardar as paredes, as janelas, as portas e o telhado, enquanto que a classe painel solar terá de guardar coisas diferentes.

Métodos: cada uma destas classes, terá métodos diferentes uns dos outros, mas existem dois métodos que todas deverão ter, que são a função de **criar** e **apagar** o objeto. Estas duas funções serão diferentes dependendo da classe.

Tendo em conta, que todas as classes "filho" terão uma estrutura idêntica, achou-se relevante enunciar apenas uma das mais importantes a **Classe Edifício**.

Classe Edifício: classe responsável por gerar edifícios.

Variáveis:

- **Estrutura completa:** guarda todas as formas geométricas que constituem o edifício.
- **Paredes do Edifício:** variável que guarda as paredes do edifício, que serão na prática um cubo.
- **Base do edifício**
- **Lista de janelas:** lista que guarda as janelas do edifício
- **Lista com caixilharia das janelas**
- **Lista de portas**
- **Outros componentes que um edifício pode ter…**

Métodos: esta classe é responsável por gerar todos os edifícios que vão aparecer no jogo, por isso terá um método para criar cada tipo diferente de edifício.

- **Apagar edifício**
- **Criar Habitações nível 1,2 e 3:** recebe como parâmetros de entrada os modelos 3D dos objetos usados na construção do edifício (como portas, janelas, etc…), a altura do edifício e a sua posição no mapa.
- **Criar Fábricas nível 1,2 e 3:** idêntico ao método criação de habitações.
- **Criar Escolas nível 1,2 e 3:** idêntico ao método criação de habitações.
- **Noite:** método que ilumina as janelas dos apartamentos entre outras coisas.
- **Dia:** método que apaga luz dos apartamentos entre outras coisas.

Classe Estrada: esta classe é bastante importante pois como foi descrito anteriormente, todos os objetos/blocos da cidade têm de estar conectados ao edifício gestor da cidade, por estradas. Esta é a classe responsável, por determinar se um objeto está ligado ao edifício gestor da cidade, ou não.

Variáveis:

- **Estrada:** variável que guarda a geometria e textura da estrada
- **A posição x e z da estrada:** duas variáveis do tipo inteiro que guardam a coordenada x e z da estrada no mapa
- **Conectado:** variável do tipo booleano que indica se a estrada está ligada ao **edifício gestor da cidade**(**EGC**)
- **Distância:** variável do tipo float que indica a sua distância relativa, ao EGC. Por exemplo se a estrada está ao lado do EGC a sua distância é igual a 0, enquanto que a estada ao lado desta tem distância 1. Caso uma estrada não esteja conectada ao EGC então fica com distância igual a infinito.
- **Vetor Apontador:** vetor de 4 posições onde cada uma representa uma posição: cima, baixo, direita e esquerda. O vetor é do tipo booleano e serve para indicar caso a estrada esteja conectada ao EGC, qual o melhor caminho para lá.
- **Seta:** variável que guarda uma seta em 3D que aponta para onde a variável anterior estiver apontada.

Métodos:

- **Construtor:** a estrada é inicializada como se não estivesse ligado ao EGC.
- **Criar:** método que para além de criar a estrada, atualiza as rotas das outras estradas para o EGC.
- **Apagar:** método que para além de apagar a estrada atualiza as rotas das outras estradas para o EGC.

Classe Cálculos do Jogo: classe responsável por calcular todos os valores que o jogador vai ter de se preocupar, como a **Saúde da Cidade**, **Felicidade da Cidade** e valor dos **recursos da cidade**.

Variáveis: variáveis desta classe vão ser os indicadores que foram descritos nas secções anteriores.

- **Saúde da cidade**
- **Felicidade da cidade**
- **Pontos de Educação**
- **Dinheiro total da cidade**
- **Balanço energético da cidade por turno**
- **Desemprego**
- **Balanço Monetário por turno**
- **Recursos Hídricos da cidade**
- **Lixo na cidade**
- **Qualidade de Ar da cidade**
- **Outros Recursos da Cidade…**

Métodos:

- **Gestão de recursos:** método principal da classe que percorre uma lista com todos os objetos da classe "Bloco jogável" criados pelo jogador (os edifícios da cidade) e calcula o estado da cidade, ou seja, a felicidade a saúde e o balanço dos recursos da cidade.
- **Alterar Saúde da Cidade**
- **Escrever valores em texto:** método utilizado para escrever num texto visível ao jogador o que foi calculado no método gestão e recursos.
- **Paga:** método que altera o valor da variável "**dinheiro total da cidade**".

Classe Menu: classe responsável por controlar todos os botões que o jogador pode interagir

Variáveis: esta classe terá uma variável por cada botão que exista no jogo, e outra variável booleana correspondente a cada botão, que indica se foi selecionado pelo jogador ou não.

Métodos: esta classe só tem um método que vai ser executado por todos os botões, que se chama "clique". Ou seja, sempre que o jogador clica num botão, é executado este método da classe Menu, que essencialmente vai verificar qual o botão selecionado e vai colocar a variável booleana correspondente a esse botão a Verdadeiro. Depois por exemplo a classe principal vai verificar que um botão X foi pressionado e depois de executar a ação correspondente ao botão pressionado, volta a colocar a variável booleana do botão X a Falso.

Classe Base De Dados: classe usada para guardar os diferentes valores de custo e produção e de consumo de cada bloco do jogo. Por exemplo, uma habitação consome e produz coisas diferentes de uma fábrica ou uma habitação de nível 2. Esta classe guarda esses valores diferentes para cada tipo de objeto/bloco que o jogador pode criar na sua cidade. Também é usada para guardar e escrever os textos informativos do jogo.

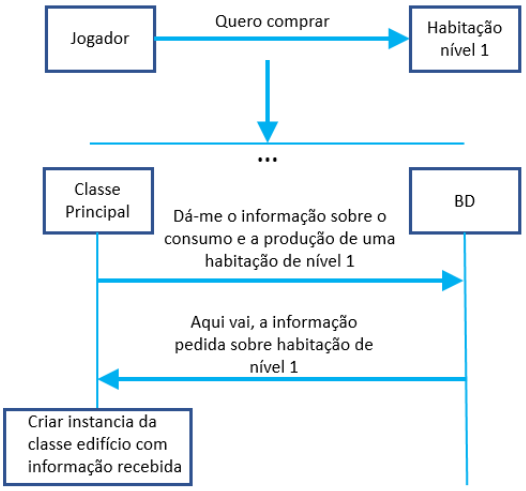


Figura 12 - Exemplo de funcionamento da classe BD

Variáveis:

- **Vetor com consumo e produção:** tal como a classe "Bloco jogável" esta classe tem um vetor semelhante. É uma variável pública de modo a ser acessível às classes que pretendam aceder a esta informação associada a um certo bloco.
- **Custo:** variável do tipo inteiro que guarda o custo de um certo tipo de bloco/objeto da cidade. Tal como no caso anterior esta variável é pública, pelas mesmas razões.
- **Educação:** variável pública, do tipo inteiro, que guarda a quantidade de pontos de educação que uma escola produz.

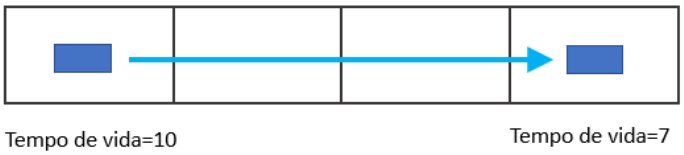
Métodos: existe um método diferente, para cada objeto/bloco diferente que o jogador pode construir. Esses métodos vão devolver a informação requisitada (vetor consumo/produção, custo e educação) para cada objeto/bloco.

Classe Movimento do carro: classe responsável pelo movimento dos carros nas estradas construídas pelo jogador. Cada objeto carro criado no jogo correrá este script, para se mover na cidade.

Variáveis:

- **Matriz de "Identificação da Cidade":** esta classe tem de receber esta matriz para o carro saber onde deve andar (onde há estrada).
- **Coordenadas x, y e z do próximo destino escolhido:** o carro para se mover na cidade terá de escolher várias vezes o próximo destino. Esta variável é um vetor do tipo float que guarda as coordenadas x, y e z do próximo destino a ir.
- **Próxima direção:** é um vetor do tipo booleano de 4 posições, que indica a próxima direção escolhida (cima, baixo, direita e esquerda).

- **Tempo de vida:** variável do tipo inteiro, que indica o tempo que o carro se move antes de estacionar e desaparecer, a próxima imagem explica melhor:



  Quando a variável chega a zero então estaciona e desaparece para poder aparecer outro carro noutro lado. Optou-se por fazer os carros desaparecer para não sobrecarregar o jogo.
- **Velocidade:** variável do tipo *float*, que determina a velocidade que o automóvel se move.
- **Coordenadas do carro:** é um vetor do tipo *float* que contém as coordenadas (x, y e z) da posição real do carro no mapa.

Métodos:

- **Mover:** método responsável por fazer mover o carro.
- **Tomar decisão:** método responsável por escolher a próxima direção a ir (cima, baixo, esquerda ou direita).
- **Estacionar:** método responsável por fazer a animação de estacionamento do carro.
- **Chegou:** método responsável por perceber quando o carro chegou ao destino.

Classe Tutorial: classe responsável por controlar o funcionamento de uma caixa de texto que indica ao jogador como se joga. Esta caixa de texto muda conforme o menu de jogo escolhido pelo utilizador. Na Figura 13 está um exemplo do seu funcionamento:

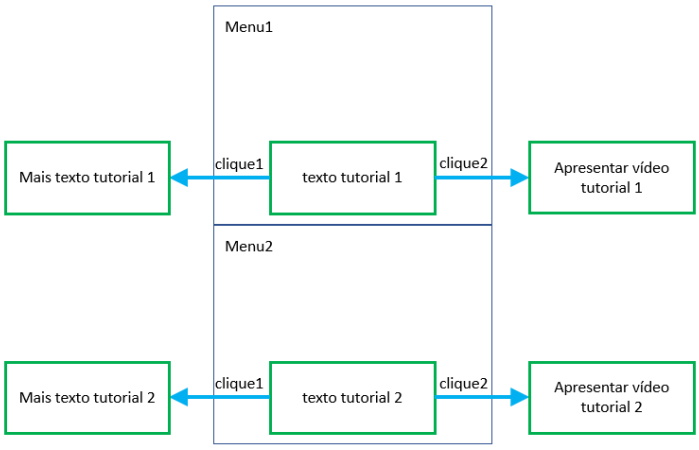


Figura 13 - Exemplo de funcionamento do texto tutorial

Variáveis: existirá uma variável para cada menu existente no jogo. Essas variáveis serão responsáveis por controlar como a caixa de texto se comporta, dependendo do menu que o jogador escolheu.

Métodos: só vai existir um método para além do construtor, esse método é executado sempre que se clica na caixa de texto. Essa função controla o comportamento da caixa de texto que depende do menu escolhido, e da interação com a caixa de texto.

Classe Movimento do Sol: classe responsável por controlar o movimento do sol e de saber se está de dia ou de noite. O tempo neste jogo tem relevância pois no final de cada turno (em cada amanhecer) calcula-se se a cidade construída é sustentável ou não através do cálculo dos indicadores como **Felicidade da Cidade**, **Saúde da Cidade**, etc.

Variáveis:

- **Velocidade:** variável do tipo *float* que indica a velocidade de rotação do sol.
- **Dia:** variável do tipo booleano responsável por indicar se está de dia ou de noite.

Método:

- **Atualizar:** esta classe tem apenas um método que corre uma vez em cada *frame* do jogo, e é responsável por fazer rodar o sol no jogo a uma certa velocidade.

Classe Mail: classe responsável por enviar emails com informação do jogo. Serve para obter informação de como o jogador jogou o jogo, podendo assim avaliar o jogo desenvolvido.

Métodos:

- **Enviar mail com anexo:** envia mail com dados sobre o jogo, enviado em anexo um ficheiro de texto com a informação e o ficheiro usado pelo jogo para carregar um jogo guardado.
- **Verificar conexão da Internet:** método que verifica se o dispositivo está ligado à internet ou não.

3.5.3 Funcionamento Geral do Jogo

Este subcapítulo, pretende explicar resumidamente o funcionamento geral da **Classe Principal** que é que vai controlar o funcionamento do jogo.

Variáveis:

- **Matriz "Cidade"**
- **Matriz "identificação da cidade"**
- **Lista de objetos da classe "Bloco jogável"**
- **Lista de objetos da classe "Estrada"**
- **Objeto da classe Menu**
- **Objeto da classe "Cálculos do jogo"**
- **Objeto da classe "Base de Dados"**
- **Inteiro com o tamanho da matriz Cidade**
- **Uma variável para cada modelo 3D usado no jogo**
- **Uma variável para cada texto visível ao jogador**
- **Uma variável para guardar o cubo vermelho descrito anteriormente**

Métodos:

- **Começar:** método que inicializa as variáveis, tais como a matriz cidade, entre outras, antes de começar o jogo.
- **Atualizar:** método que corre uma vez por cada imagem gerada. É o método que gere o funcionamento total do jogo.
- **Um método para criar cada objeto/bloco que o jogador pode construir**
- **Editar objeto/bloco**
- **Apagar objeto/bloco**
- **Criar Carros:** sempre que se cria um carro, este desaparece passado algum tempo. Este método serve para criar numa posição aleatória do mapa (onde haja algum objeto) um carro e para gerir a quantidade de carros que existe na cidade.
- **Novo Jogo:** método responsável por criar o terreno inicial do jogo.
- **Carregar Jogo:** método responsável por recriar o mapa criado e guardado pelo jogador
- **Dia:** método responsável por gerir efeitos da cidade durante o dia, por exemplo apagar as luzes da cidade ou ativar efeito de partículas dos painéis solares, etc…
- **Noite:** método responsável por gerir efeitos da cidade durante a noite, por exemplo ligar as luzes da cidade ou desativar efeito de partículas dos painéis solares, etc…
- **Fim do jogo:** método responsável por realizar tarefas de fim do jogo, como escrever "Fim do Jogo" e voltar ao menu inicial.

Fluxograma do funcionamento Geral Do Jogo:

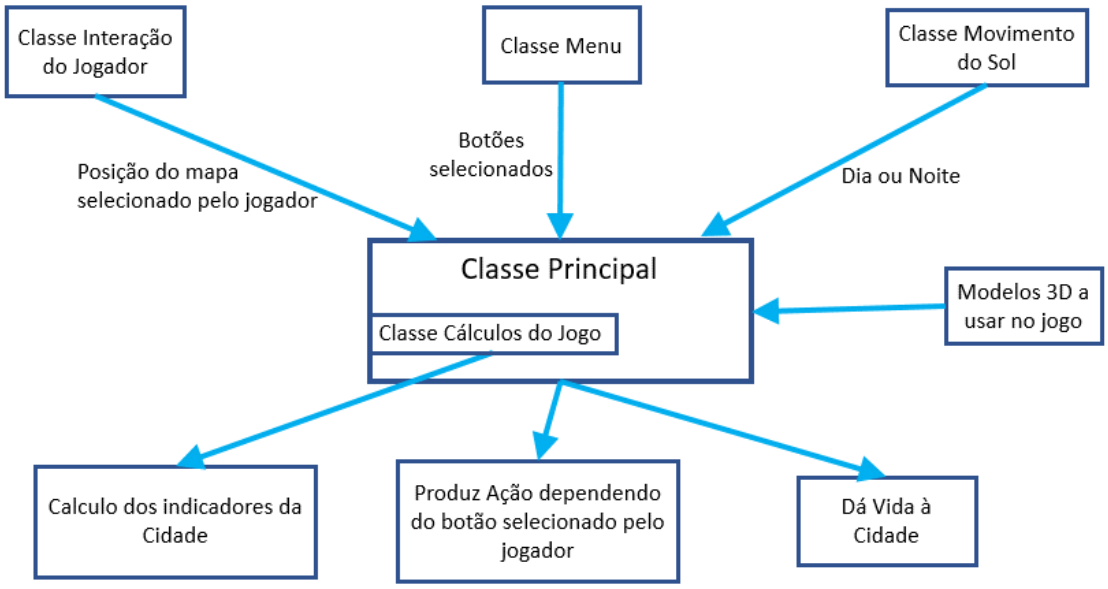


Figura 14 - Diagrama do funcionamento geral do jogo

Na Figura 14 encontra-se um diagrama que apresenta um sumário sobre o funcionamento geral do jogo. A classe principal recebe como input informação das outras classes e produz resultados a partir deles, como calcular valores como a Saúde da Cidade, Felicidade da Cidade, etc. A partir dos menus selecionados pelo jogador a classe Principal irá construir edifícios, estradas, terreno, etc; e por fim também serve para dar vida a cidade, ou seja, gerir a iluminação da cidade, construir carros, entre outros.

Fluxograma do funcionamento da classe Principal, ou seja, do método "Atualizar":

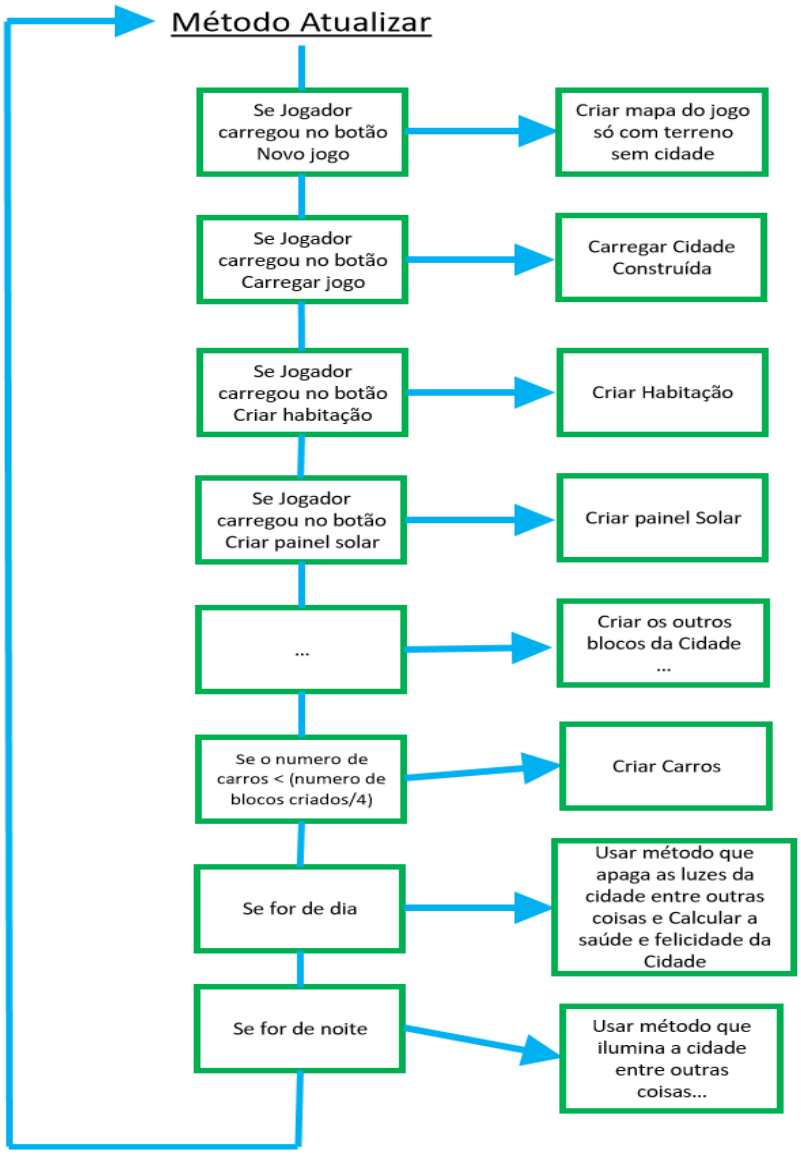


Figura 15 - Fluxograma do método atualizar

Na Figura 15 é apresentado o que a classe principal executa em cada imagem processada do jogo. Na Figura 16, pretende-se expor o diagrama de classes UML do programa, que se pode ver na próxima página.

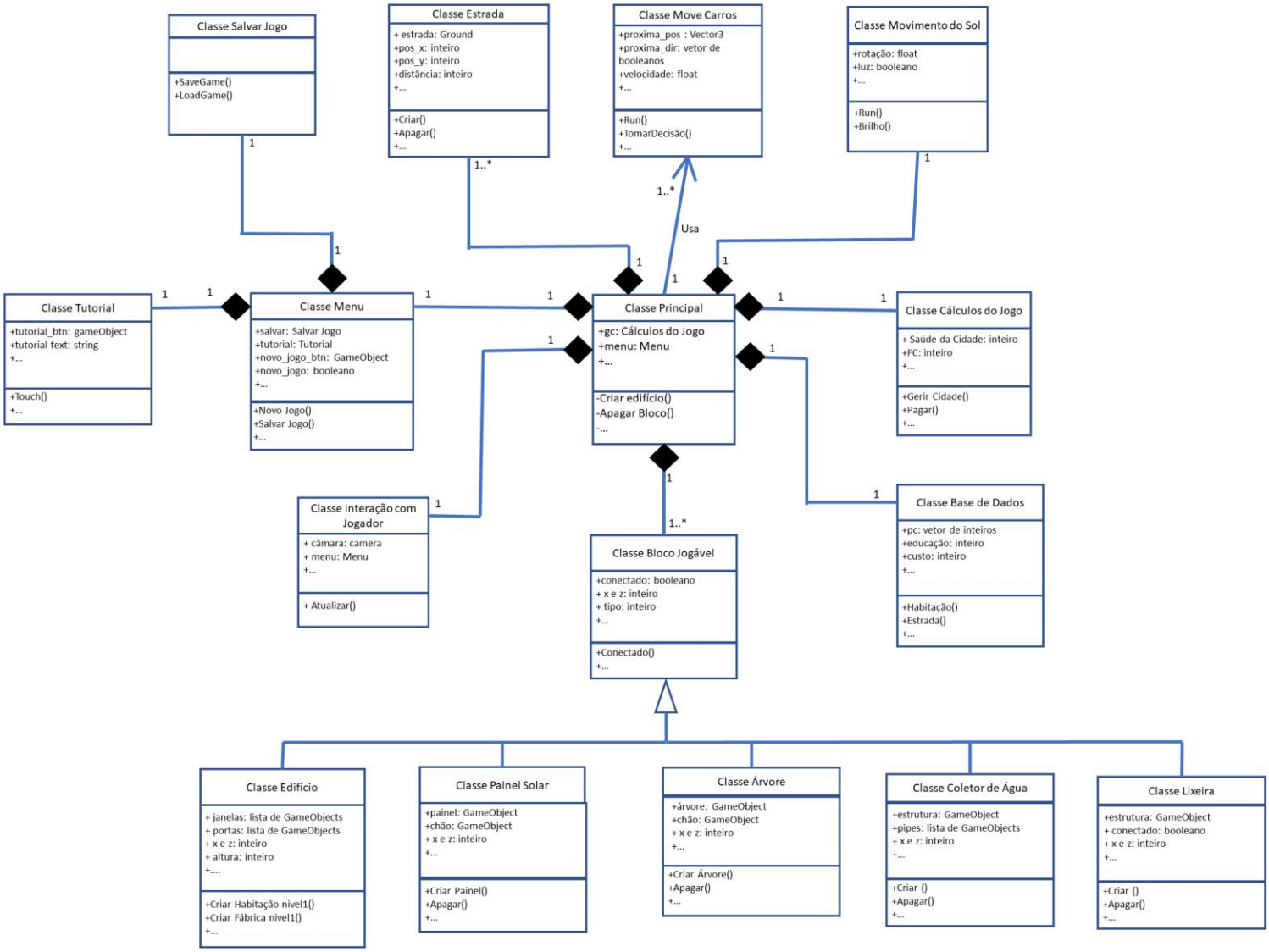


Figura 16 - Diagrama de Classes UML

3.6 RESUMO

O jogo a realizar pretende cumprir os seguintes objetivos, de modo a distinguir-se de jogos semelhantes e resolver o problema que esta dissertação pretende cumprir, que é **ensinar através de vídeo jogos os conceitos básicos e fundamentais sobre as SC**, tais como:

1. O problema que levou à criação do conceito das SC.
2. Ensinar como a obtenção de informação (Big Data, IoT), ou melhor conhecimento, sobre a cidade, através do uso de IoT, pode ajudar a fazer uma gestão mais inteligentes dos recursos da cidade.
3. Explicar a importância do meio ambiente e da educação nas SC.

Para cumprir o primeiro objetivo, propõe-se a realização de um vídeo introdutório do jogo, a explicar este assunto. Para cumprir o segundo requisito, propõe-se o uso de indicadores visíveis ao jogador, que o informem sobre o estado que se encontra a cidade. Também pretende-se explicar como a cidade obteve esses dados, através do vídeo inicial e da construção e ligação dos diversos componentes constituintes da cidade, ao edifício gestor da cidade e através de diversos textos informativos, espalhados pelos menus do jogo. Por fim, para explicar a importância do meio ambiente e educação, vai-se contar, no primeiro caso (meio ambiente), com a mecânica central do jogo explicada na secção 3.2. Em relação à educação nas SC, a dependência de pontos de educação para obter melhores versões dos blocos constituintes da cidade, pretende resolver esta questão.

Outros requisitos que se pretendem validar nesta dissertação são uns mais relacionados com a dimensão de jogabilidade. Um primeiro requisito a cumprir é que o jogo deve ser **fácil de instalar e começar a jogar**. Deve estar acessível para jogar no **maior número de plataformas possível**. Deve ser **simples de jogar**, para que crianças possam jogar, mas com alguma dificuldade para ser desafiante. O jogo deve ter uma **curva de aprendizagem rápida** e deve ser **divertido de jogar**. Por fim, o jogo deve dar ao jogador alguma liberdade na forma de jogar e deve ter boa longevidade. O jogo proposto neste capítulo já pretende cumprir estes requisitos, mas estes serão validados com mais detalhe no próximo capítulo.

4 VALIDAÇÃO

O objetivo deste capítulo é, apresentar e explicar o protótipo desenvolvido e analisar os resultados obtidos. Em primeiro lugar vai-se apresentar o protótipo e todas as suas funcionalidades, de seguida vai-se explicar como foi desenvolvido o jogo, descrevendo como foram resolvidos os maiores desafios no seu desenvolvimento. Por fim, serão apresentados os resultados obtidos, validando de seguida os requisitos anunciados no capítulo anterior.

4.1 PROTÓTIPO DESENVOLVIDO

Este subcapítulo, pretende descrever como ficou o protótipo do jogo desenvolvido, e ver que diferenças teve da proposta feita no capítulo 3.

4.1.1 Menu Inicial

Figura 17 - Menu inicial

Ao começar o jogo, o jogador depara-se com este menu. Tem-se duas hipóteses, ou fazer "*New Game*" para começar um jogo novo, ou "*Load Game*" para carregar um jogo gravado previamente. Ao clicar no "*Load Game*" o ecrã poderá congelar uns segundos (1-2 segundos) enquanto começa a carregar (construir) a cidade gravada previamente. Ao carregar o jogo, não existe uma representação visual do tempo de espera do mesmo, pois achou-se que a sua implementação não teria importância, já que o tempo médio de carregamento é de apenas alguns segundos.

Figura 18 - Menu de escolha de dificuldade

Se o jogador escolher começar um novo jogo, então vai-se deparar com o menu da Figura 18. Este menu é composto por 5 botões e duas barras horizontais. Começando pelos três botões principais: "Easy"," Normal"," Hard" são os três modos de dificuldade no jogo. Como foi referido na proposta do jogo, a cidade terá vários indicadores relevantes para o jogador saber o estado da cidade. Dois dos mais importantes indicadores (ou medidas) são o "Dinheiro Total da Cidade" e a "Saúde da Cidade". Usando as duas barras horizontais deste menu é possível escolher um valor, maior ou menor, para estes indicadores. Por exemplo, caso o jogador escolha o modo de jogo Fácil, então a cidade começará com dinheiro máximo e saúde no máximo, caso escolha modo difícil, o contrário. O nível de dificuldade, também altera o custo de pontos de educação necessários para comprar as versões melhoradas de cada bloco. Também existe a hipótese de o jogador escolher um nível personalizado de dificuldade, se mexer no valor das barras horizontais. Por fim, existe um botão para retornar ao menu inicial.

4.1.2 Início Do Jogo

Ao começar um novo jogo, é apresentado imediatamente um pequeno vídeo (1 minuto e 18 segundos) a explicar o problema que as SC estão a tentar resolver e como pretendem fazer isso.

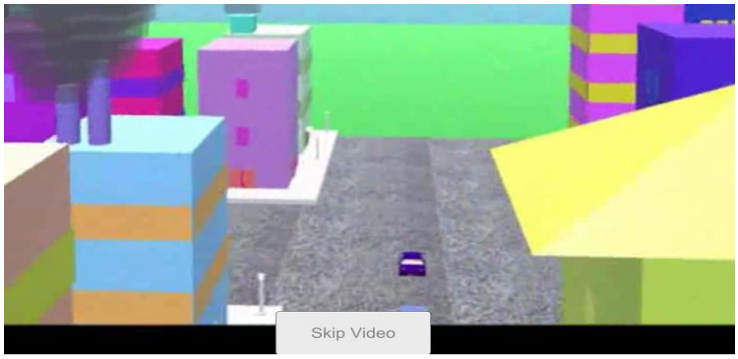


Figura 19 - Imagem retirada do vídeo introdutório do jogo

Na Figura 19 é possível ver um recorte do vídeo inicial que passa no jogo. O jogador tem a hipótese de saltar o vídeo, caso não o queira ver. Depois de ver o vídeo o jogador depara-se com o seguinte cenário:

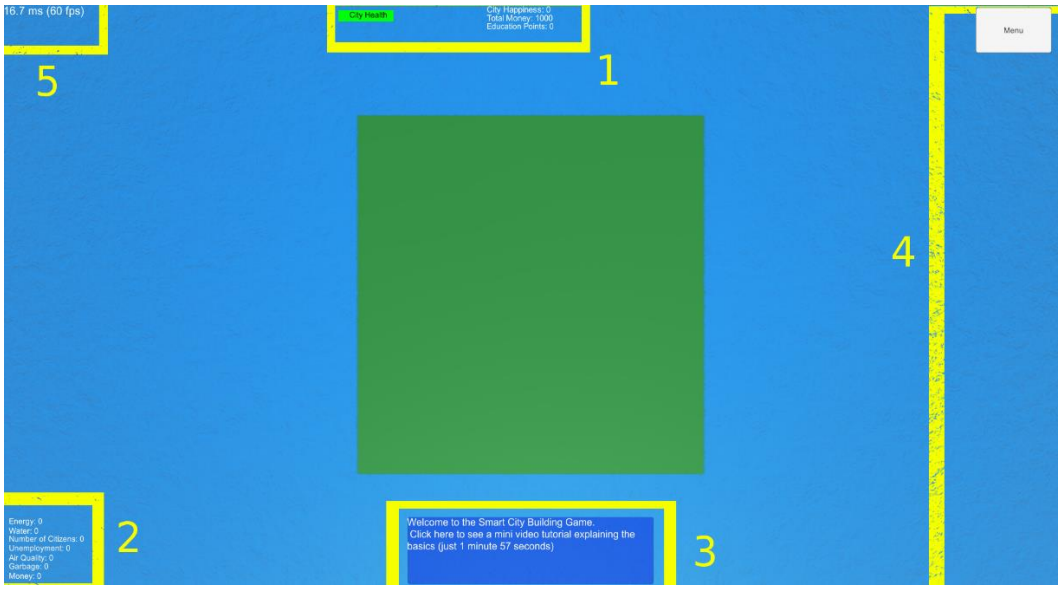


Figura 20 - Cenário no início de um novo jogo

Como é possível observar na Figura 20, o mapa do jogo é constituído por algumas zonas, a verde é o terreno onde o jogador poderá construir a sua cidade (pode-se aumentar esta zona se o jogador quiser), a azul o mar ou oceano e as zonas a amarelo com números, são os menus e textos, que o jogador pode ver (os retângulos e números a amarelos, não aparecem no jogo). Passando a descrever as regiões numeradas:

1. Nesta secção é possível ver uma barra horizontal verde com um texto a dizer “City Health” ou “Saúde da Cidade” e um texto que indica três medidas importantes no jogo como foi descrito na proposta: “Felicidade da Cidade”, “Dinheiro Total da Cidade” e “Pontos de Educação”. Cada um destes medidores vem acompanhado de um número a indicar o seu valor.
2. Nesta área são apresentados os valores associados aos recursos da cidade. Os recursos considerados no jogo são: **energia, água, numero de habitantes, desemprego, qualidade do ar, lixo e dinheiro**. Os valores destes recursos podem variar entre positivo e negativo, sendo negativo se existir falta do recurso, ou positivo caso exista abundância. Estes valores variam sempre no final de cada turno e dependem dos blocos que o jogador criou em cada turno do jogo. Alguns dos recursos são especiais, como o dinheiro e o desemprego. O dinheiro nesta secção indica se a cidade está a ganhar ou a perder este recurso por turno. O valor de desemprego é um dos mais importantes pois, caso seja negativo (ou seja, existe emprego, não existem trabalhadores) ou positivo (existe trabalhadores, não existe emprego), a cidade fica a perder. Portanto, ter este valor a zero é o ideal. Por fim, a soma destes valores exceto o número de habitantes,

será o valor de “Felicidade da Cidade” que por sua vez tem impacto direto no valor da “Saúde da Cidade” que determina se o jogador perde ou não.

3. Texto que serve de tutorial. O jogador pode clicar neste texto para abrir pequenos vídeo tutoriais ou para ler mais texto informativo, de como se joga o jogo.
4. Esta área é onde o jogador, vai interagir com os menus principais do jogo. No início só existe um botão “Menu” que ao clicar nele, abre todos os outros menus necessários para jogar. Este botão é importante pois nos subsequentes submenus existirá um botão “Return” que permite voltar para este menu inicial. Os submenus que existem são os seguintes: opções, navegação, comprar blocos (habitações, fábricas, painéis solares, etc.), controle do tempo de cada turno e por fim um botão para editar blocos.
    a. **Botão de Opções**, ao clicar, abre várias outras opções tais como: ligar ou desligar os textos tutoriais, ligar ou desligar a música do jogo, voltar para o menu inicial, sair do jogo, salvar a cidade construída, para mais tarde poder carregá-la novamente, e finalmente, ligar ou desligar o modo de conexão, que permite ver se os blocos estão conectados ao edifício gestor da cidade ou não.
    b. O botão de **navegação** serve para o jogador puder navegar pelo mapa do jogo.
    c. O submenu “**Comprar Blocos**” permite ao jogador comprar e colocar no mapa, o bloco comprado, também é aqui que se pode apagar do mapa um bloco. Como foi escrito previamente, cada bloco custa X de dinheiro, caso o jogador não tenha dinheiro para comprar um bloco, ele pode construir o bloco na mesma, mas depois fica com dívida a pagar, que subtrai diretamente na “Saúde da Cidade”. Nem todos os blocos estão disponíveis para comprar no início, alguns custam “Pontos de Educação”, que a cidade pode obter se tiver escolas.
    d. O submenu **Controle do Tempo** permite ao jogador controlar a velocidade que demora cada turno do jogo.
    e. Por fim, o botão **editar blocos** serve só para aumentar o número de andares de cada edifício. Por exemplo uma habitação de dois andares conta como se tivessem duas habitações no mesmo sítio. Infelizmente não se conseguiu adicionar um botão para criar a versão melhorada de um bloco, ou seja, a única maneira que o jogador tem de criar uma habitação de nível 2 num sítio onde já exista uma habitação de nível 1 é apagando o bloco e criado o bloco de nível 2 novamente na mesma posição.
5. Este não é muito importante, serve apenas para ver o número de imagens por segundo, que o jogo gera, é um indicador relevante para perceber se o jogo está a correr fluentemente ou não.

Estes são os elementos essenciais do jogo. Vamos passar para uma pequena demonstração de jogabilidade.

4.1.3 Decorrer Do Jogo

Ao começar o jogo, como foi visto anteriormente, é possível escolher o nível de dificuldade, se o jogador escolher o modo fácil, a cidade começa com 500 unidades de "Saúde da Cidade" e com 1000 unidades de "Dinheiro da Cidade".

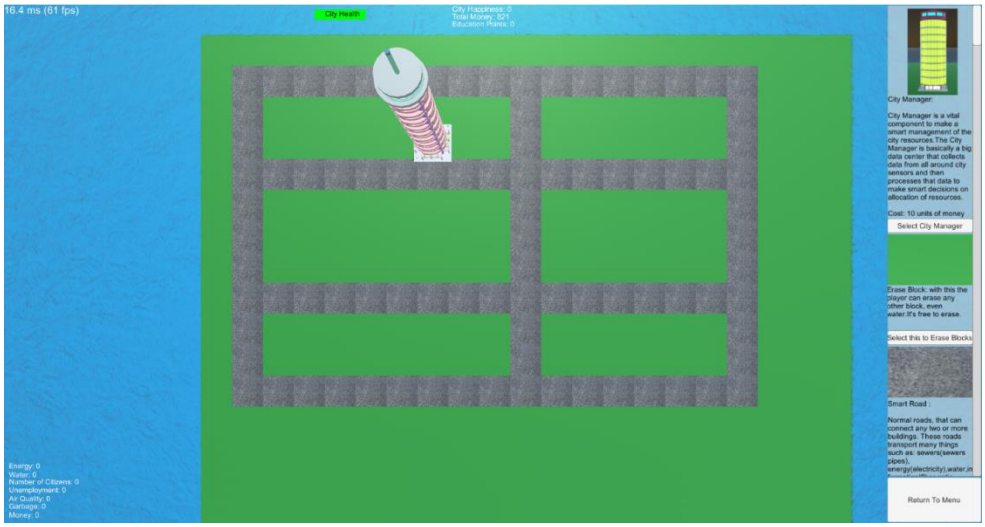


Figura 21 - Início de construção de uma cidade no jogo

Como se pode ver na Figura 21, já se começou a construir as estradas da cidade. Já existe um edifício, que é o Gestor da Cidade(EGC), que se falou na proposta. Para construir alguma coisa no jogo é necessário primeiro construir este edifício gestor como foi explicado previamente. Na figura talvez não seja possível ver, mas o **dinheiro total da cidade** já não é 1000 como foi iniciado, mas 821, devido à construção dos elementos presentes na imagem. À direita na imagem encontra-se um painel com todos os elementos que o jogador pode comprar para a cidade, sendo que alguns deles só podem ser comprados através do consumo de pontos de educação. Cada elemento neste painel tem um texto explicativo, que explica conceitos sobre as SC e também indica o que consome e produz cada elemento no painel, como foi visto na proposta.

Figura 22 - Cidade de Dia

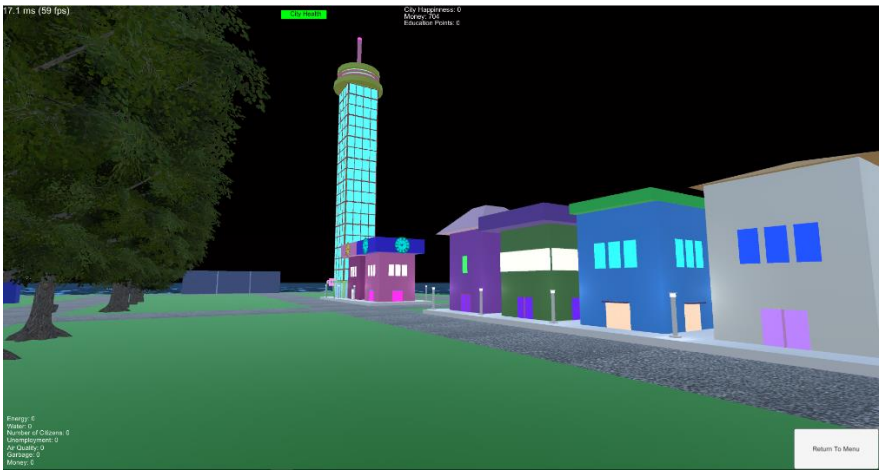


Figura 23 - Cidade de Noite

Cada turno do jogo é limitado no tempo, ou seja, cada turno começa com o amanhecer do sol e termina no dia seguinte, no novo amanhecer (por volta de 1 minuto, se o jogador não alterar a duração do turno). Para o jogador ter tempo de pensar o que vai construir, existe a hipótese de parar o tempo. Como é possível observar nas figuras, a cidade muda ligeiramente de noite e de dia, por exemplo de noite as luzes da cidade acendem-se e os painéis solares deixam de absorver luz (que é representado por um efeito de partículas). Os turnos são importantes no jogo pois, no final de cada turno, calcula-se se a cidade é sustentável ou não (calcula-se os recursos que a cidade produz e consome). Nestas figuras também é possível observar que o jogador pode mover a câmara livremente no jogo e navegar pelo mapa da cidade.

Figura 24 - Cidade construída no final do primeiro turno

Figura 25 - Zoom da Figura 24 na parte de indicadores do topo

Figura 26 - Zoom da Figura 24 na parte dos recursos

Como se pode ver na Figura 24, a cidade já está mais composta, existem habitações (número 3 da Figura 24), extratores de água para a cidade (2), árvores (4), painéis solares (5), lixeiras (6), fábricas (7) e escolas/centros de pesquisa (1). Os dados que aparecem na Figura 25 e Figura 26 não são aleatórios, estão associados com a mecânica central do jogo, onde cada elemento da cidade consome e produz algo. Passando a explicar como funciona a mecânica proposta no capítulo anterior agora na prática:

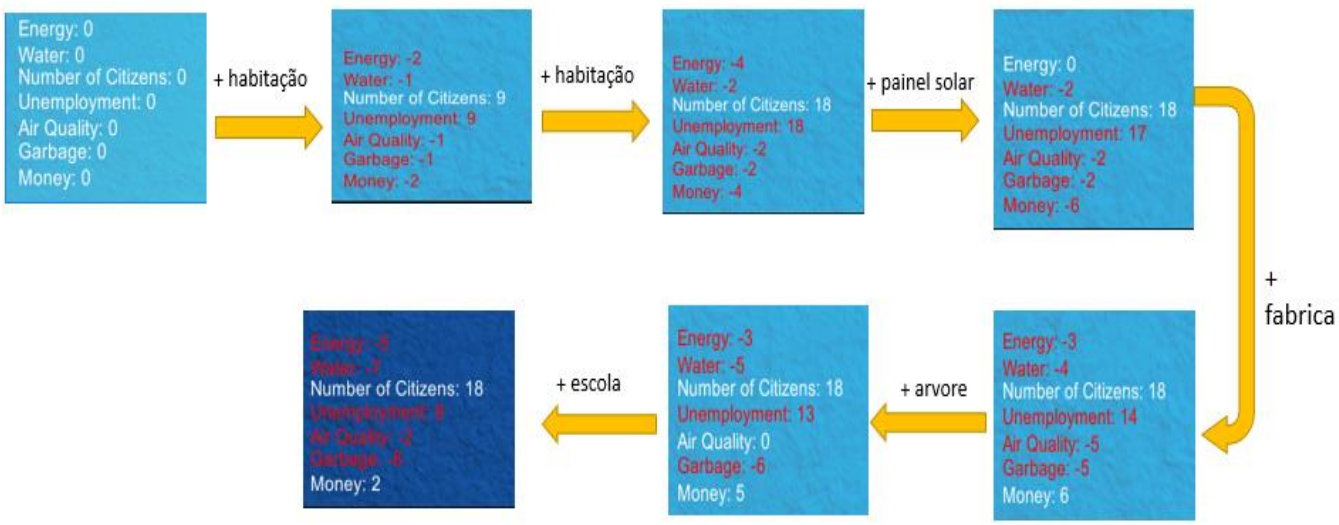
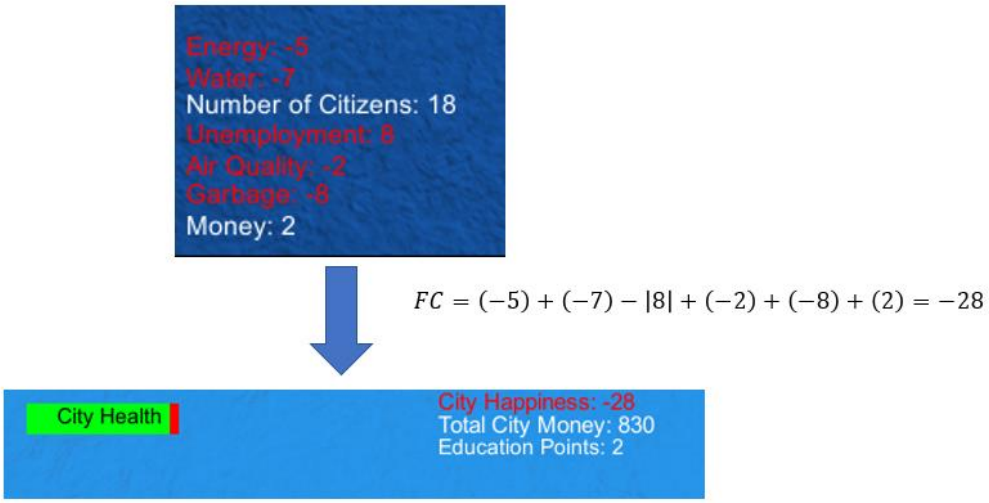


Figura 27 – Exemplo da mecânica central do jogo

Como se pode ver na Figura 27 (um exemplo diferente da Figura 24) os recursos da cidade começam todos a zero, pois não foi construído ainda nada. O edifício EGC e as estradas não alteram estes recursos. Ao se criar uma habitação na cidade, verifica-se que os recursos mudaram, a cidade passou a consumir duas unidades de energia uma de água e duas unidades de dinheiro, por turno. Uma habitação também produz 9 habitantes, uma unidade de lixo e uma unidade de poluição. Ao adicionar mais uma habitação esses valores duplicam, mas se adicionarmos um painel solar então o balanço energético da cidade já fica a zero, ou seja, toda a energia produzida pelo painel solar é consumida. Esta imagem torna claro o funcionamento desta mecânica do jogo. O jogador vai vendo os recursos atualizarem-se à medida que vai construindo, mas é no final do turno que se tem as consequências da cidade construída.



$$SC_t = SC_{t-1} - |D| + FC + \min(DTC, 0)$$

Figura 28 - Cálculo da Felicidade da Cidade e Saúde da Cidade

No final do turno, o jogo calcula o valor da **Saúde da Cidade**, os **Pontos de Educação** e **Felicidade da Cidade** com base nas fórmulas indicadas na Figura 28, que serão explicadas mais à frente neste capítulo. Como foi indicado no capítulo anterior, o valor da **Saúde da Cidade** vai

determinar se o jogador perde ou permanece no jogo, daí a sua importância. A fórmula da **Saúde da Cidade** é a indicada como $\mathbf{SC_t}$ que significa **Saúde da Cidade** no turno presente. A **Saúde da Cidade** do turno presente é igual ao do turno anterior menos o módulo do **desemprego** (D), mais a **Felicidade da Cidade** (FC) mais o mínimo do valor **Dinheiro Total da Cidade** (DTC) até zero, ou seja, só quando DTC<0 é que participa na equação negativamente. Chegou-se a esta fórmula para penalizar os jogadores que não se interessam com o desemprego, ou se endividam muito. Para o jogador se preocupar com os outros recursos, este tem de cuidar o valor da **Felicidade da Cidade** (FC) que como se pode ver na Figura 28 é a soma de todos os recursos apresentados ao jogador, com exceção do desemprego que caso seja positivo ou negativo, afeta negativamente a FC, portanto o objetivo é ter este valor a zero ou perto dele. Em relação aos **Pontos de Educação**, estes são calculados da seguinte forma: cada escola criada produz e consome recursos, mas as escolas são especiais, pois produzem um recurso especial que chamaremos de **pe**, que entra na fórmula que calcula os **Pontos de Educação**(**PE**) que o jogador vê.

$$PE_t = PE_{t-1} + \frac{\sum_{i=1}^{nº\ total\ de\ escolas} pe_i}{(\frac{nº\ de\ cidadãos}{4})}$$

Explicando a fórmula anterior, os **Pontos de Educação** do turno presente são iguais aos pontos do turno anterior mais a soma de todos os **pe's** produzidos pelas escolas a dividir por um quarto do número total de cidadãos. Dividiu-se por este valor para dar a noção ao jogador que quanto maior a população mais escolas o jogador tem de criar para obter os mesmos pontos de educação.

Outra questão importante que o jogador necessita se preocupar é em ligar a cidade por estradas, vejamos um exemplo em concreto:

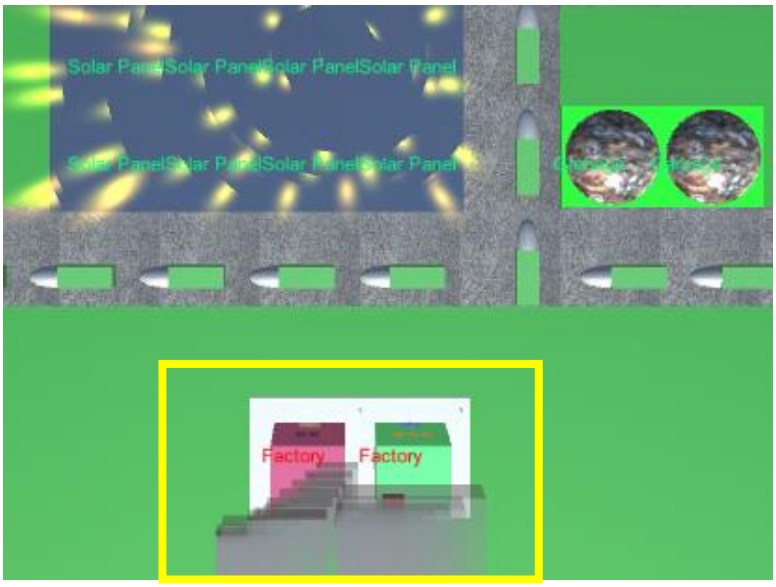


Figura 29 - Exemplo de blocos não ligados ao EGC

Na Figura 29 podemos observar (dentro da caixa amarela) que duas fábricas não estão ligadas, pois não têm nenhuma estrada diretamente ligada a elas. Caso isto aconteça estes blocos não vão participar como recursos na cidade, ou seja, não produzem nem consomem.

Concluindo, o objetivo do jogador é criar uma cidade que seja sustentável, ou seja, que não perca **Saúde da Cidade** em cada turno e tenha o maior valor de **Felicidade da Cidade** possível. Para isso o jogador vai ter de se preocupar necessariamente com recursos como a qualidade do ar e o lixo produzido, entre outros. Os **Pontos de Educação**, tem extrema importância, pois o jogo foi testado de modo a que seja necessário comprar as versões melhoradas dos edifícios para tornar a cidade sustentável. O jogo está feito de modo a que seja possível o jogador recuperar **Saúde da Cidade** caso a tenha perdido.

4.1.4 Detalhes

Todos os elementos disponíveis para comprar no jogo são: **edifício gestor da cidade, ruas, habitações, escolas, fábricas, painéis solares (fonte de energia limpa), fábrica que produz energia de forma poluente (pode representar qualquer fonte de energia poluente), bombas de água (para recolher água para a cidade), lixeiras e árvores**. Em termos de manipulação de terreno, pode-se criar **terreno, água(mar) e praias**.

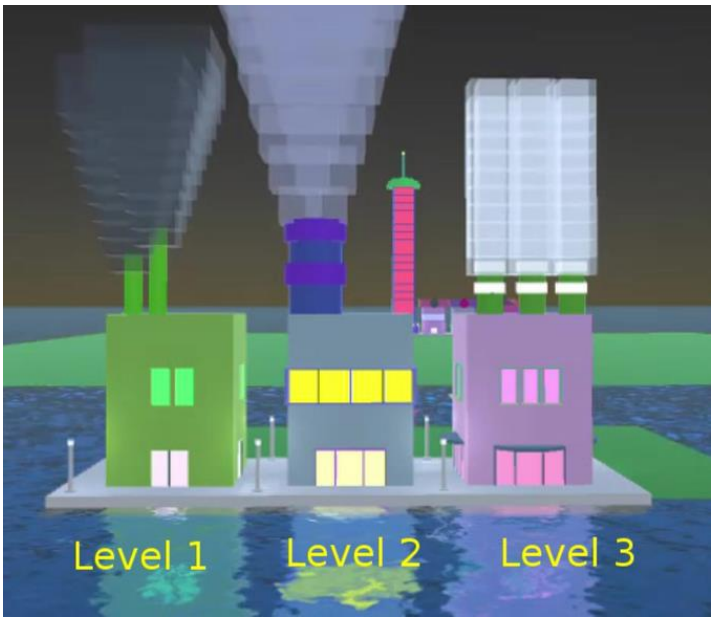


Figura 30 - Vários níveis das fábricas

Quase todos os componentes (exceto árvores e as opções de manipulações de terreno) no jogo têm 3 versões ou níveis diferentes como foi mencionado previamente. Na Figura 30 pode-se ver isso, cada nível muda o aspeto e a produtividade de cada elemento.

4.1.5 Recolha De Dados Do Jogador

Quando o jogador deseja sair do jogo, aparecem duas janelas, uma a dizer se quer enviar os dados do jogo e se quer responder a um questionário, a outra diz “Exit” para fechar o jogo. Se o jogador quiser responder ao questionário online, então clica na janela e abre-se uma página de internet para um “Google Form”. O formulário/questionário usado, encontra-se no capítulo 4.3. Para além disso o jogo envia um email, com um ficheiro de texto, com informação dos Dados do jogo, como a “Saúde da Cidade”, “Felicidade da Cidade”, recursos… e por fim também envia em anexo um ficheiro com a cidade construída, ou seja, se alguém carregar este ficheiro no jogo consegue ver a cidade construída por essa pessoa. Isso possibilita, mais tarde, que se possa carregar no jogo as cidades feitas pelas pessoas que responderam ao questionário.

4.2 DESENVOLVIMENTO DO JOGO

Este capítulo tem a intenção de explicar, como foi feito o jogo, ou seja, explicar os desafios superados no desenvolvimento do jogo. É necessário referir, mais uma vez, que se usou a ferramenta de criação de jogos Unity para criar o jogo presente nesta dissertação, e a linguagem de programação usada foi C#.

4.2.1 Principais Desafios

Para construir este jogo, os grandes desafios que se teve de resolver foram os seguintes:

1. Interação do Jogador com o Jogo: ou seja, como o jogador interage, de modo a movimentar-se no mapa, a perceber como se joga, como se constrói os edifícios, como se guarda a cidade construída, etc.
2. Geração de edifícios, como posso gerar edifícios diferentes uns dos outros sem ter de fazer grandes modelos em 3D? Pois tomaria muito tempo no desenvolvimento do jogo.
3. Como saber se o jogador fez uma cidade que faça sentido? Como foi explicado previamente, a solução passa por ligar todos os edifícios, por estradas, ao edifício gestor da cidade. Como detetar se os edifícios estão ligados ao gestor da cidade? Esta é a pergunta a resolver.
4. Como dar vida a cidade, ou seja, movimento? Uma das soluções é fazer o movimento de veículos na cidade. Como fazer isso?
5. Por fim como gerir os dados no jogo, como se sabe os recursos gastos?

Estes são assim os desafios mais importantes a resolver no jogo, no próximo subcapítulo será descrito como foi resolvido cada um destes problemas, sendo que alguns foram parcialmente resolvidos no capítulo 3. Também se pretende explicar como se resolveu alguns dos desafios mais secundários.

4.2.2 Resolução Dos Problemas

Sendo alguns dos desafios mais abstratos do que outros, vai-se primeiro descrever a solução encontrada, aos que são mais específicos. O terceiro é um dos mais específicos e com uma solução interessante, portanto comecemos por esse.

Como foi explicado previamente, todos os blocos do jogo, tanto de terreno, como de água (mar), edifícios, estradas, etc… estão todos guardados na **matriz Cidade** e sempre que o jogador cria um edifício no mapa, está apenas a atualizar/mudar uma das posições da matriz Cidade já previamente inicializada. Esta matriz trabalha em conjunto com outra (**matriz Identificação da Cidade**) idêntica na forma, mas que apenas guarda inteiros, esses valores representam que tipo de bloco está guardado nas coordenadas x e z do mapa. Tendo relembrado isto vejamos como foi resolvido este desafio.

A primeira solução encontrada foi, sempre que se criava por exemplo um edifício, verifica-se se existe na sua vizinhança uma estrada, se sim inicia-se um caminho pelas estradas até chegar ao EGC. Se encontrar um caminho para o EGC, então está ligado, caso contrário está desligado. O algoritmo usado para encontrar o caminho mais próximo para o gestor da cidade, foi a distância euclidiana[1], ou seja, dos 4 vizinhos disponíveis num ponto da matriz, verifica-se quais são estradas. Desses calcula-se a distância euclidiana de cada um ao edifício gestor da cidade (que é uma coordenada conhecida) e compara-se, com os outros, para ver quem tem a menor distância. O vizinho que tiver a menor distância é escolhido para ser o novo ponto de referência onde se iniciará novamente este algoritmo, até chegar ao edifício gestor da cidade. Se o ponto de referência chegar ao destino, então está conectado. O algoritmo tinha mais heurísticas para resolver este problema, mas rapidamente viu-se que não conseguia resolver este problema para todos os casos possíveis. Na Imagem seguinte vê-se o que chegou a ser feito.

Figura 31 - Primeiro algoritmo de conexão

Na Figura 31, vê-se o protótipo feito deste algoritmo, sendo o círculo amarelo o EGC, e a azul o elemento a verificar se está conectado. Este algoritmo para além de não funcionar bem para todos os casos, não era prático, visto que sempre que o jogador criava um edifício ou apagava uma estrada, tinha-se de executar novamente este algoritmo para todos os elementos da cidade.

O algoritmo encontrado para resolver esta questão para além de funcionar muito bem, também é prático e rápido. A solução passa por associar uma informação a cada módulo de estrada, essa informação é a distância a que se encontra do EGC.

[1] $\sqrt{x^2 + z^2}$ esta é a formula da distância euclidiana, mas foi usada mais uma distância como a x + z, que não obteve bons resultados.

Ao criar uma estrada, verifica-se na sua vizinhança a existência de outra com distância <∞ ao EGC. Se encontrar, iguala-se a sua distância ao EGC com o vizinho que tenha a menor distância, e soma-se uma unidade a esse valor. Caso não encontre nenhuma estrada, então coloca a sua distância igual a infinito. Este algoritmo já é bastante eficaz, visto que ao criar uma habitação, ao lado de uma estrada, pode-se verificar imediatamente a distância, associada a essa estrada, e se for menor que infinito, significa que está ligado ao EGC. Parece simples, mas não basta isto para funcionar. O jogador tem a capacidade de criar e eliminar estradas livremente, então como verificar a conexão de uma cidade que está sempre a modificar-se? Para isso é necessário um algoritmo de propagação de informação, ou seja, sempre que se cria ou se elimina uma estrada, é necessário que esse bloco que foi apagado por exemplo, indique aos vizinhos, que por ali já não existe um caminho válido. Portanto, a partir de agora vai-se descrever o pseudo código usado na classe estrada para resolver esta questão.

Como foi explicado no capítulo 3, a classe Estrada tem algumas variáveis e métodos que ajudam a resolver este desafio, tais como a posição (x e z) a distância ao EGC e o vetor que aponta para o melhor caminho. De seguida, será mostrado como são inicializadas algumas destas variáveis:

- Distância = infinito
- Vetor apontador= new bool [4] { false, false, false, false };

Depois de inicializado usa-se o método - **Criar** para construir um bloco de estrada e atualizar a ligação dos blocos de estrada vizinhos caso necessário. Vejamos o seguinte exemplo:

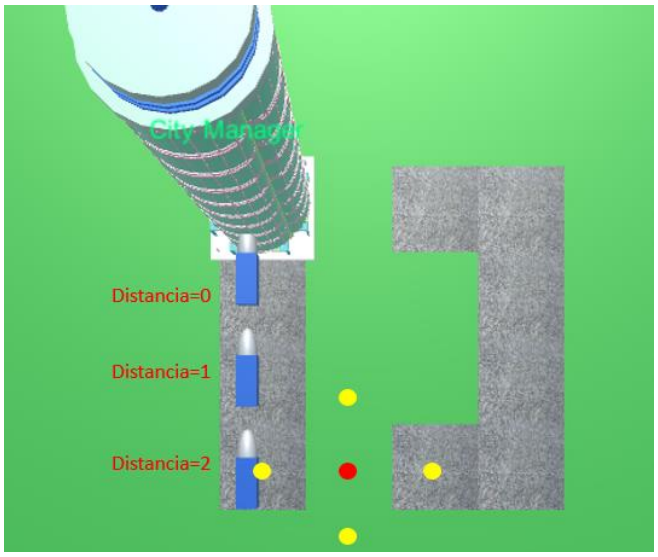


Figura 32 - Exemplo da classe Estrada a funcionar

Como se pode ver na Figura 32 deseja-se criar uma estrada no ponto vermelho, e os pontos amarelos representam os vizinhos desse bloco. Tendo em mente este exemplo vai-se descrever o algoritmo geral usado tanto pelo método **Criar** como **Apagar**, as diferenças serão explicadas de seguida.

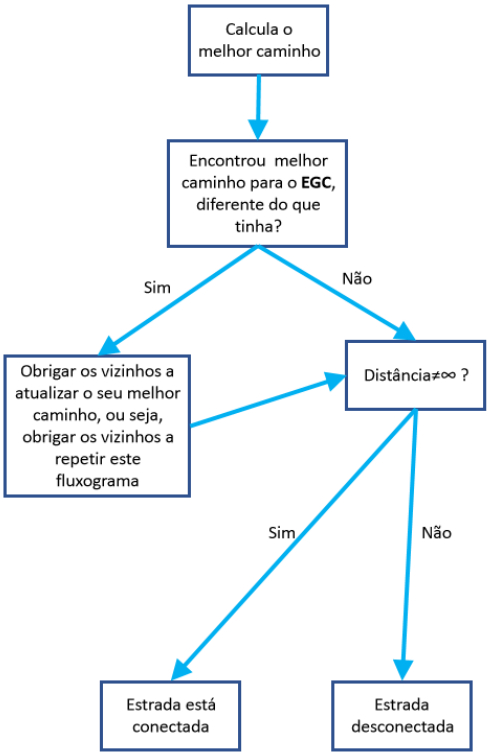


Figura 33 - Fluxograma do funcionamento da classe Estrada

De forma resumida, este é o algoritmo usado para saber quais as estradas conectadas ou não ao EGC. O método **Criar** faz exatamente o que está descrito na Figura 33**,** ou seja, começa sem estar a apontar para lado nenhum e com a variável distância igual a infinito. Depois, calcula o novo melhor caminho e prossegue exatamente como se mostra no fluxograma. No método **Apagar,** apaga-se a estrada e obriga-se os vizinhos dessa estrada a fazer este fluxograma. Depois de criar uma estrada no ponto vermelho da Figura 32, fica assim:

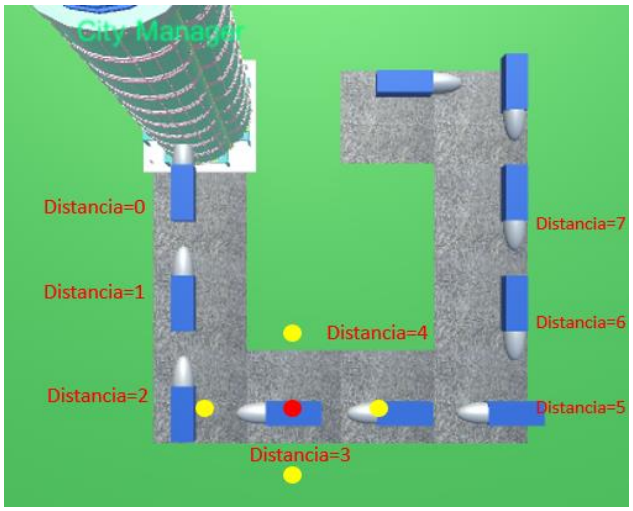


Figura 34 - Demonstração da classe estrada a funcionar

Com este algoritmo não existe problema de se ficar num ciclo infinito, ou seja, ficar numa situação, onde cada estrada obriga os vizinhos a atualizar a sua distância sem parar. Isso

acontece porque existe uma condição de paragem. A condição de paragem é que uma estrada só deve obrigar os vizinhos a calcular o melhor caminho caso, a estrada ao calcular o seu melhor caminho, obtiver um caminho diferente ao que tinha anteriormente. Assim, a propagação de informação costuma estabilizar rapidamente. Este algoritmo não comete erros (tipo ***loop***) como o que se pode ver na Figura 35, e também consegue encontrar o caminho mesmo que seja muito longo, que o algoritmo antigo (o que não funcionou) também não fazia.

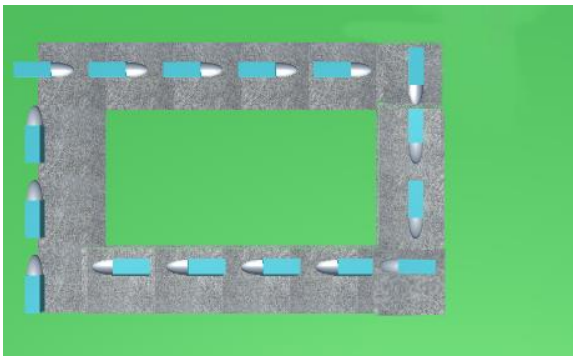


*Figura 35 - Exemplo de **loop***

Outro problema mais direto de responder é o de como dar vida à cidade? A solução encontrada passa por dar movimento a vários elementos da cidade e fazer um sistema de tráfego na cidade, resumindo:

1. Iluminação na cidade
2. Dar movimento aos diversos elementos presentes na cidade
3. Sistema de tráfego.

Como foi descrito no capítulo 3, a classe **Edifício** tem variáveis e listas com cada componente do edifício. Também existem dois métodos desta classe **Dia** e **Noite**, que tratam de iluminar e apagar as luzes dos edifícios. Por exemplo, o método **Noite** da **Classe Edifício** percorre todos os objetos que se devem iluminar, como as janelas ou os candeeiros e ilumina-os. Para fazer isso, transforma-se a cor (do objeto a iluminar) de RGB para HSV, e depois altera-se o valor de brilho do HSV (o ***value***) e volta-se a converter essa cor para RGB e aplica-se no material. Também se altera uma propriedade do material que se chama **emissão**. Esta propriedade **emissão** que os materiais do **Unity** têm, permite que os objetos difundem essa cor para o ar dando efeito de estar iluminado. O contrário acontece no método **Dia**. Depois, no método **Atualizar** da **Classe Principal**, é só necessário saber quando é de noite ou de dia, e depois é só preciso percorrer todos os edifícios da cidade e usar este método **Noite** ou **Dia**, em cada edifício**.**

Para dar movimento aos diversos elementos da cidade, pensou-se no seguinte. Os edifícios não se podem mexer, portanto não vale a pena, dar-lhes movimento. Árvores também não, mas as bombas de água, já vale a pena dar algum movimento como o de sucção. Para imitar um movimento parecido ao de sucção, fez o seguinte: as bombas de água têm um código a correr que contém dois estados, um para aumentar a largura e diminuir a altura, e outro para encolher a largura e aumentar a altura. Para controlar quando se altera de um estado para outro, usa-se o tempo, como medida de alteração de estado, por exemplo de 2 em 2 segundos muda-se de estado (ver código nos anexos). Os painéis solares e fábricas também deveriam ter algum tipo de movimento. Nos painéis solares foi usado um sistema de partículas que o Unity fornece.

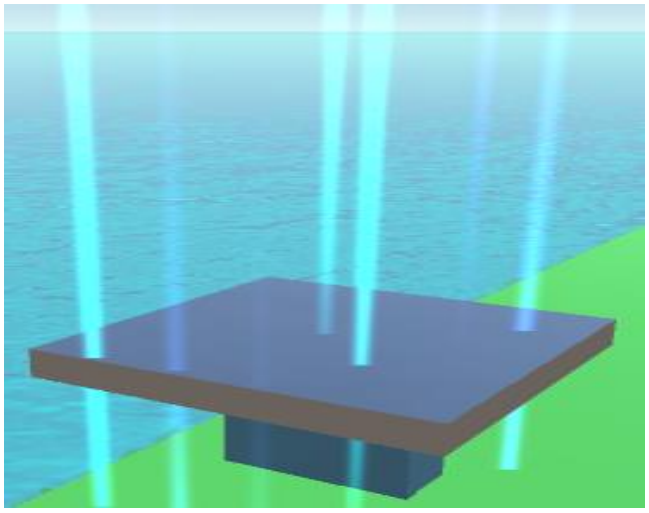


Figura 36 - Sistema de partículas no painel solar

Para edifícios do tipo fábrica, criou-se chaminés, que têm um sistema de partículas a fazer de fumo. Mais uma vez, existe um código em cada chaminé, que funciona do seguinte modo, de x em x segundos é criado uma partícula de fumo. São criadas partículas de fumo até chegar ao limite máximo de partículas que se decidiu dar, para não sobrecarregar a performance do jogo. Em cada iteração do programa, o código percorre todas as partículas de fumo e altera a sua posição numa direção aleatória vertical.

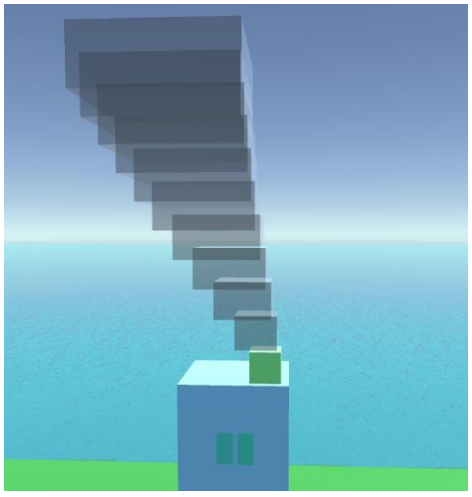


Figura 37 - Chaminé de uma fábrica

Como se pode ver na Figura 37, o fumo está a ir numa direção. De x em x segundos, o fumo muda de direção. À medida que o fumo vai subindo a partícula de fumo também altera o seu tamanho na seguinte proporção: no eixo **X** e **Z** cresce **W** no eixo **Y** cresce **W/4**(ver código nos anexos).

Por fim o último movimento relevante é a dos edifícios que absorvem energia solar.

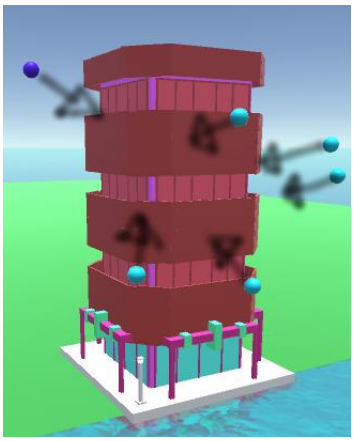


Figura 38 - Edifício a absorver energia solar

Este sistema de partículas funciona do seguinte modo, criam-se (no centro do edifício) esferas de duas cores diferentes e adiciona-se uma distância aleatória a essa partícula (para afastá-las do edifício). Depois, um ciclo passa por todas as partículas criadas e calcula-se a diferença entre a sua posição atual e a final (no centro do edifício), depois soma-se uma percentagem pequena dessa diferença (0.01%) à posição atual da partícula, de modo que as partículas vão diminuindo a velocidade quando se aproximam do destino e acelerem quando estão longe. À medida que as partículas chegam ao destino, são destruídas e são criadas novas. Este sistema tem mais detalhes importantes na sua implementação, mas não são relevantes para esta dissertação (ver código nos anexos).

Agora que falamos de alguns dos elementos mais importantes que tornam a cidade viva, vejamos o sistema de tráfego da cidade. Cada carro que anda na cidade tem um código que lhe faz andar pelas estradas da cidade. Para não ter muitos carros a andar na cidade, determinou-se que seria uma boa quantidade, se o número de carros a circular na cidade fosse sempre 1/4 do número de elementos da cidade (como edifícios, painéis solares, árvores, …). Assim só se cria novos carros na cidade, enquanto o número de carros for menor que o (número de elementos da cidade) /4. Tendo em vista as variáveis e os métodos da classe que controla o movimento dos carros, enunciados no capítulo 3 vejamos como ficou o fluxograma do funcionamento (de alto nível) do código responsável por mover o carro pela cidade.

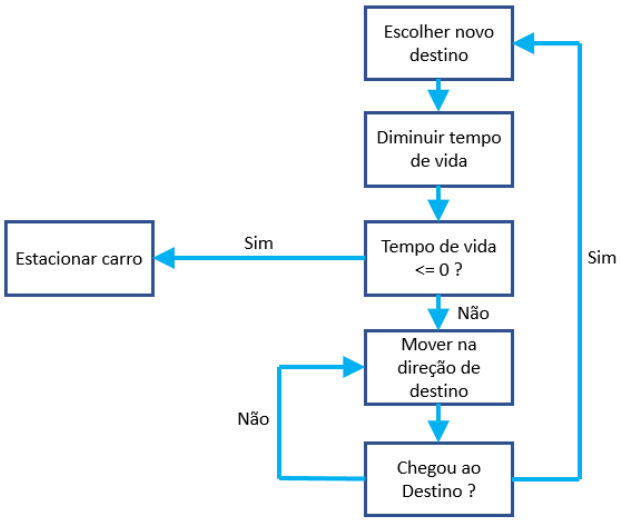


Figura 39 -Fluxograma do funcionamento de alto nível do movimento de um carro

Do fluxograma da Figura 39, vale a pena explicar as seguintes caixas: **escolher novo destino**, **mover na direção do destino** e **chegou ao destino**. Comecemos pelos mais fáceis de explicar

Chegou ao Destino:

```
bool arrivedNextPos(float distance)
{
    bool arrived;
    float x, z;
    x = Mathf.Abs(this.transform.position.x - nextpos.x);
    z = Mathf.Abs(this.transform.position.z - nextpos.z);

    if (x <= distance && z <= distance)
    {
        arrived = true;

    }
    else
    {
        arrived = false;
    }
    return arrived;


}
```

Figura 40 - Código da função chegou ao destino

Como se pode ver na Figura 40 para ver se o carro chegou ao destino, basta calcular o módulo da diferença entre a posição atual (x e z) e a posição destino e verificar se é menor que um valor **W**, que neste caso é 0,05.

Mover na direção do destino:

```
void Move(bool moveup, bool movedown, bool moveright, bool moveleft)
    {
        if (moveup)
        {
            this.transform.localEulerAngles = new Vector3(0,-90,0);
            this.transform.position = new Vector3(this.transform.position.x, this.transform.position.y, this.transform.position.z + Time.deltaTime * speed);
        }
        if (movedown)
        {
            this.transform.localEulerAngles = new Vector3(0, 90, 0);
            this.transform.position = new Vector3(this.transform.position.x, this.transform.position.y, this.transform.position.z - Time.deltaTime * speed);
        }
        if (moveright)
        {
            this.transform.localEulerAngles = new Vector3(0, 0, 0);
            this.transform.position = new Vector3(this.transform.position.x + Time.deltaTime * speed, this.transform.position.y, this.transform.position.z);
        }
        if (moveleft)
        {
            this.transform.localEulerAngles = new Vector3(0, 180, 0);
            this.transform.position = new Vector3(this.transform.position.x - Time.deltaTime * speed, this.transform.position.y, this.transform.position.z);
        }
    }
```

Figura 41 - Código da função mover

Como se pode ver na Figura 41 o código também é muito simples, a função recebe como parâmetro de entrada a direção para onde o carro se deve mover (cima, baixo, esquerda ou direita). Em cada um dos casos, primeiro altera-se a rotação do carro no eixo do Y e de seguida altera-se a posição do carro de acordo com a seguinte fórmula: seja $\boldsymbol{p}(t) \in \mathbb{R}^3$ a posição atual do carro. A próxima posição $\boldsymbol{p}(t + dt) = \boldsymbol{p}(t) + dt\boldsymbol{v}$, onde $\boldsymbol{v} \in \mathbb{R}^3$ é a velocidade do carro.

Escolher novo destino:

Este método apesar de simples é mais comprido, e é responsável por escolher a direção a seguir de modo a não sair da estrada.

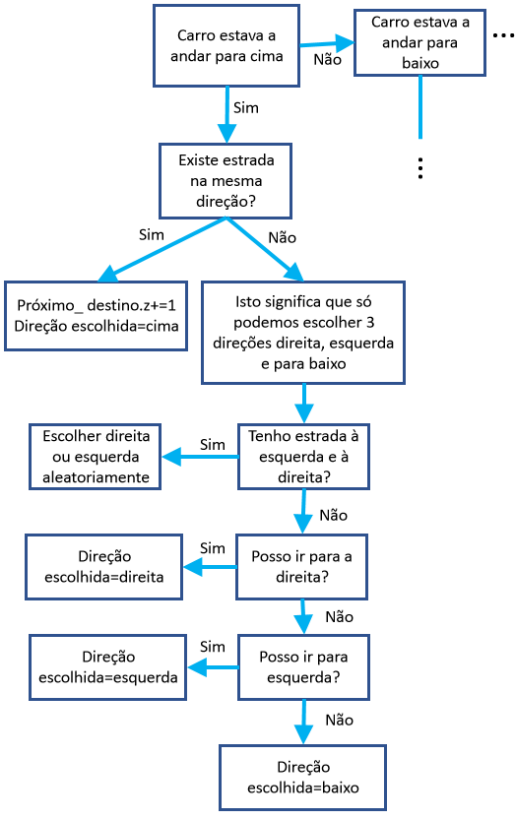


Figura 42 – Fluxograma da função escolher novo destino

Na Figura 42 só é apresentado uma parte da função **escolher novo destino**, as restantes são muito semelhantes. As partes que faltam são as condições se o carro estava a andar para a esquerda, direita ou para baixo. Todas funcionam de forma idêntica só as direções é que trocam.

4.2.3 Geração De Edifícios

Sentiu-se a necessidade de criar uma ferramenta de gerar edifícios pelas seguintes razões:

1. Para simplificar a criação de edifícios, pois criar à "mão" modelos 3D de edifícios é muito difícil e trabalhoso.
2. Para dar mais liberdade na criação de edifícios. No jogo pretende-se dar a liberdade ao jogador para construir edifícios de qualquer altura, para isso é necessário criar uma ferramenta destas. Para além disso, os modelos retirados da internet poderiam ficar mal no jogo, visto que podem ter níveis de detalhe diferentes uns dos outros, entre outros problemas.

Para simplificar a implementação desta ferramenta, cada edifício da cidade ocupa uma posição na matriz do mapa, tal como os restantes componentes da cidade. No início cada edifício começou por ser apenas um cubo (estrutura principal) com larguras constantes e alturas diferentes, pois o Unity permite fazer isso facilmente. Como isto é demasiado simplista, decidiu-se dividir os edifícios em algumas partes essenciais, como a base, a estrutura principal (o cubo

principal), a entrada (portas), janelas e por fim o telhado. Deste modo bastaria criar modelos simples de portas telhados, janelas, etc… e combinar os diferentes modelos para ter edifícios relativamente diferentes.

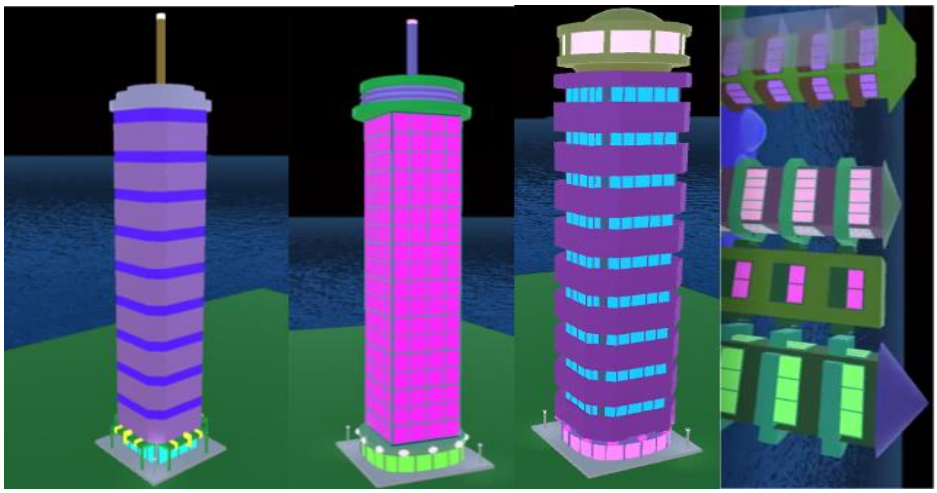


Figura 43 - Geração de edifícios

Como se pode ver na Figura 43, existe relativa diversidade nos edifícios construídos, usando apenas uma combinação de alguns modelos diferentes de janelas e cores diferentes.

Para se explicar com mais detalhe a construção de edifícios é relevante mostrar qual o procedimento constante (na classe Principal) na construção de cada objeto que o jogador pode criar. Para isso vai-se mostrar o procedimento para um caso concreto, mas todos os outros são muito semelhantes.

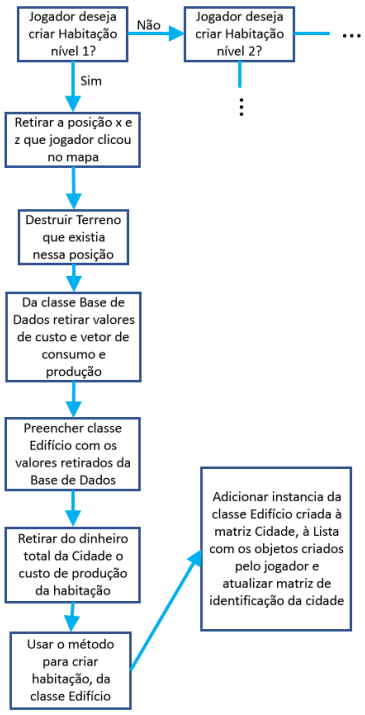


Figura 44 - Procedimento (alto nível) da criação de uma habitação

O método **Criar** da classe Edifício, recebe como parâmetros de entrada da função os vários modelos que pode usar.

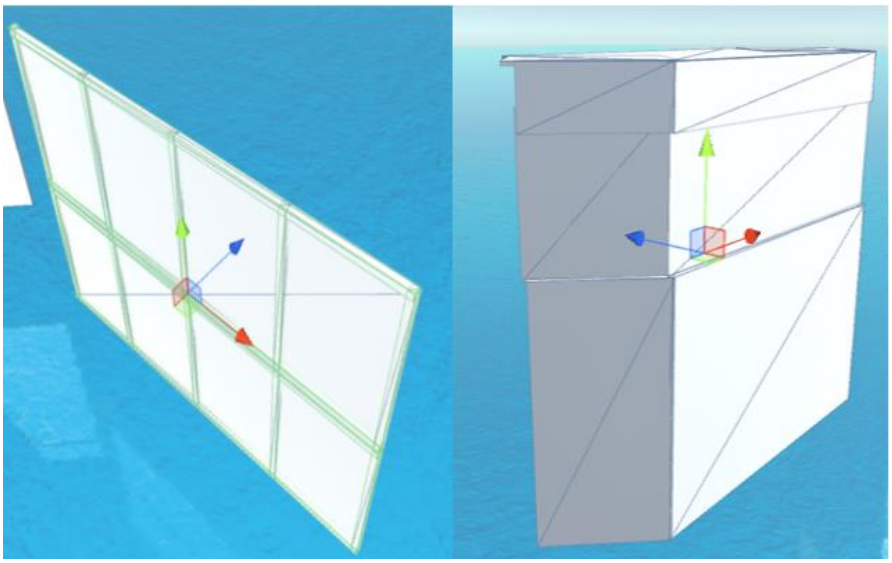


Figura 45 - Alguns dos modelos 3D das janelas usadas

Sempre que se cria um edifício, escolhe-se aleatoriamente um dos modelos enviados nos parâmetros de entrada da função (janelas, portas e telhados), e escolhe-se também aleatoriamente cores para inserir nos modelos usados nos edifícios (como o cubo principal, janelas, portas, etc…) como se vê na Figura 45. Para colocar os modelos no cubo principal do edifício, é preciso rodar os modelos e deslocá-los de modo a ficarem colados nas faces do cubo. Para dar liberdade ao jogador de construir edifícios de alturas diferentes, arranjou-se um algoritmo para calcular a posição que cada modelo de janela tem, conforme a altura do edifício (ver código nos anexos).

4.2.4 Gestão De Dados Do Jogo

A classe responsável por gerir os dados que o jogador visualiza é a classe **Cálculos do Jogo,** em particular, o método principal **Gestão de Recursos**. Esta função é responsável por calcular o valor da Saúde da Cidade e de todos os outros valores que permitem o seu cálculo. Para isso esta função tem acesso à lista que contém todos os blocos criados pelo jogador. Ao executar esta função, a primeira coisa a fazer é percorrer a lista e somar o vetor de consumo e produção de cada objeto num único vetor.

Cada edifício/bloco da cidade tem um vetor com o seguinte formato:

$$[energia,\ \ água,\ \ cidadãos,\ \ qualidade\ do\ ar,\ \ recolha\ do\ lixo,\ \ dinheiro]$$

Como foi visto anteriormente cada valor deste vetor pode tomar valores positivos (no caso de produzir) ou negativos (no caso de consumirem o recurso). Para somar tudo num vetor:

$$\begin{bmatrix} energia_{objeto\,1} & \cdots & energia_{\,objeto\,n} \\ \vdots & \ddots & \vdots \\ dinheiro_{objeto\,1} & \cdots & dinheiro_{objeto\,n} \end{bmatrix} \times \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} =$$

$$= \begin{bmatrix} balanço\ energético\ da\ cidade \\ \vdots \\ balanço\ monetário\ da\ cidade \end{bmatrix} (1)$$

Com o vetor (**1**) obtemos, o balanço dos recursos da cidade que o jogador vê no canto inferior esquerdo do ecrã. A partir dele é possível calcular todos os outros indicadores visíveis para o jogador. A "Felicidade da Cidade", como foi explicado previamente é a soma de todos estes valores em (**1**). O dinheiro total da cidade é calculado, somando o balanço monetário em (**1**) à variável que guarda o dinheiro total da cidade na classe Cálculos do Jogo. A "Saúde da Cidade" calcula-se assim: $SC_t = SC_{t-1} - |D| + FC + \min(DTC, 0)$ onde **FC** é "Felicidade da Cidade" e $\boldsymbol{DTC}$ representa "Dinheiro Total da Cidade", **D** é o desemprego, $\boldsymbol{SC_{t-1}}$ representa a "Saúde da Cidade" no turno anterior.

4.2.5 Interação Jogador Com O Jogo

Por fim vejamos, como foi feito a interação com o jogador. Como foi escrito previamente, tentou-se fazer com que a interação do utilizador com o jogo fosse o mais táctil possível, para funcionar bem tanto no computador como em dispositivos móveis. Por isso a interação que o jogador faz é através de muitos menus onde o jogador pode clicar. Decidiu-se dividir a interação do utilizador nas seguintes funções:

1. Movimentação da câmara pelo mapa
2. Construção de objetos no mapa do jogo
3. Controlo dos botões do menu do jogo
4. Tutorial
5. Música

4.2.6 Movimentação Da Câmara Pelo Mapa

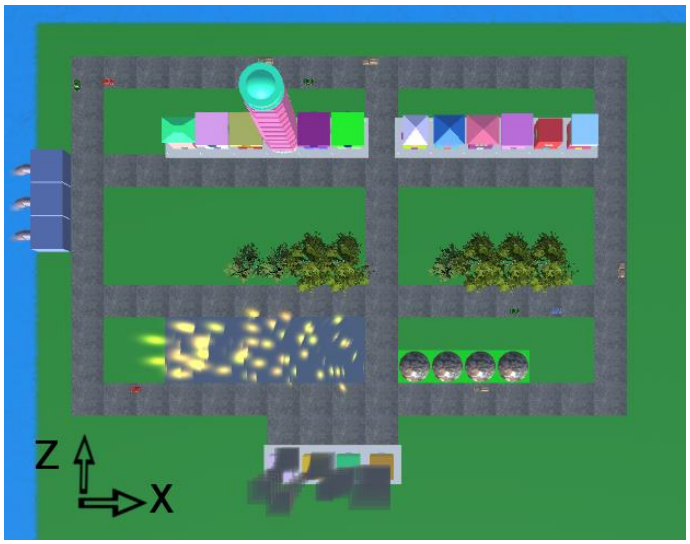


Figura 46 - Vista em planta do mapa do jogo

A vista normal do jogo é em planta. O Unity consegue detetar o movimento do cursor, usou-se isso para mover a câmara do jogo. Sempre que o jogador clica no menu que permite a navegação no mapa, este fica habilitado a mexer na câmara do jogo. Para isso, basta clicar com o botão esquerdo do rato e arrasta-lo para cima e para os lados, para mover a câmara nos eixos X e Z. Para mover no eixo Y, usa-se o botão direito do rato. Fez-se o movimento da câmara deste modo, pois pensou-se que funcionaria tanto na versão móvel como de computador. Visto que isso não aconteceu, teve-se de desenvolver outra abordagem para movimento da câmara, nos dispositivos móveis.

Figura 47 - Versão móvel do jogo

A solução encontrada é a que se vê na Figura 47, onde se usa a esfera que se encontra dentro do quadrado amarelo, para fazer de cursor. Em vez dos botões do rato, usa-se dois botões que se encontram na Figura 47, no quadrado vermelho. Um deles serve para mover a câmara nos eixos X e Z e o outro para mexer no eixo Y. Aproveita-se para falar da versão móvel do jogo. Praticamente funciona igual, à versão do computador exceto esta parte da navegação e o facto de usar os dedos em vez do teclado. O problema mais grave que impediu a conclusão desta

versão do jogo, foi a velocidade que corre o jogo nos dispositivos móveis. A placa gráfica dos dispositivos móveis é bastante mais fraca que a dos computadores, o que torna o jogo muito lento mesmo para cidades muito pequenas. Tentou-se exportar o jogo com menor qualidade gráfica e mesmo assim teve má performance. Tentou-se implementar um sistema para mudar o nível de detalhe dos edifícios, dependendo da posição da câmara.

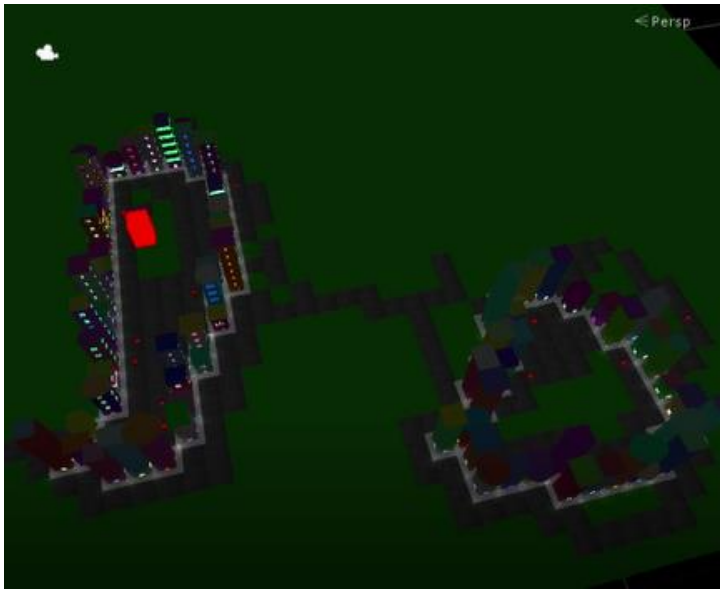


Figura 48 - Sistema de nível de detalhe

A Figura 48 mostra o sistema de nível de detalhe a funcionar. Consiste essencialmente em ocultar detalhe dos edifícios e não só, quando a câmara não incide sobre eles. Como não se obteve resultados satisfatórios, não se chegou a utilizar este sistema e desistiu-se de concluir a versão móvel do jogo.

Finalmente, em relação à navegação no jogo, existe mais um modo, que se chama modo de **câmara livre**, onde o jogador pode "voar" livremente com a câmara pelo mapa do jogo. Para se mover neste modo o jogador tem de clicar num outro botão e depois só necessita de usar as teclas **w, a, s, d, e, q, Shift, Ctrl** para mexer a câmara. Para sair deste modo basta clicar no **Esc** e no botão **Return** do menu para retornar ao menu inicial. Este modo só funciona na versão para o computador.

4.2.7 Construção De Objetos No Jogo

Para mexer no "cenário 3D" do jogo, criou-se um cubo (que é invisível ao jogador na maior parte do tempo) que é controlado pelo cursor ou dedo da seguinte forma:

Figura 49 - Clique do rato no jogo

A Figura 49 explica muito bem o que sucede no jogo. Existe um cubo no jogo que está invisível ao jogador. Esse cubo é depois usado para saber a posição, onde se deve criar, por exemplo, um edifício. Ao clicar com o rato no ecrã, cria-se um raio/linha que vai desde a câmara até ao terreno nas coordenadas (X;Z) onde se clicou com o cursor no ecrã. Deteta-se o objeto que colidiu com o raio, por exemplo o terreno, sabendo as coordenadas desse objeto, pode-se saber onde construir o edifício. Usou-se o cubo para que o jogador possa saber onde clicou e para ajudar na implementação, pois guarda a última posição que o jogador clicou, visto que a é a mesma do cubo.

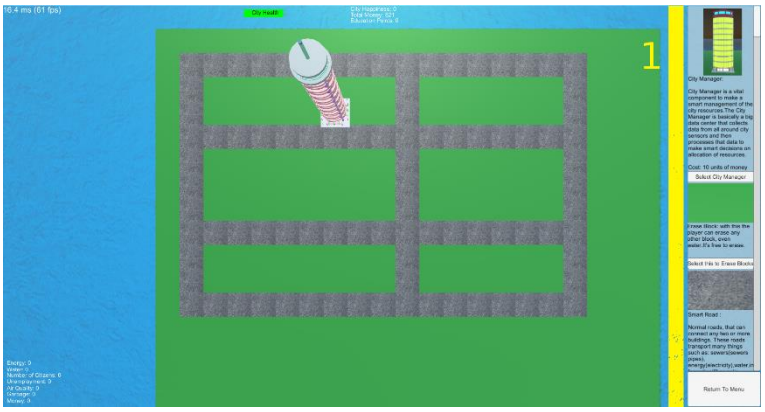


Figura 50 – Painel de Compra de Blocos.

Para criar o painel de compra de edifícios para a cidade, utilizou-se um menu deslizante que o Unity fornece. Para criar um edifício, primeiro clica-se no menu, na imagem do objeto que se deseja construir, depois clica-se no mapa numa coordenada X;Z para criar o objeto escolhido. Os textos visíveis por debaixo de cada imagem do painel estão guardados na classe Base de Dados. No início do jogo, ou seja, no método **Começar** da **Classe Principal**, é executado um método da classe **Base de Dados** que preenche com texto informativo, que ensina muito resumidamente conceitos das cidades inteligentes sobre o objeto em questão e o que produz e

consome. Assim, facilita-se a atualização desses textos e valores (produção e consumo), sempre que não se achar que estão corretos.

4.2.8 Controlo De Menu Do Jogo

Como foi referido previamente, existe uma classe que se chama **Menu** que controla todos os botões do ecrã. No fundo, deteta quando algum menu foi selecionado pelo jogador. Esta classe é também responsável por mostrar e esconder e revelar os botões certos na altura certa. Desde os botões no menu inicial, até aos do painel de compra, são todos tratados aqui.

Esta classe também é responsável por salvar o jogo. Para isso, temos que guardar a seguinte informação: que tipo de objeto se encontra em cada posição do cenário (matriz de identificação do mapa do jogo), a altura de cada edifício, dados sobre o estado da cidade (recursos, Saúde da Cidade, Dinheiro total da cidade, pontos de educação, etc…) e por fim o custo educacional de edifícios nível 2 e 3, pois mudam dependendo da dificuldade escolhida no início do jogo. Para guardar e carregar o jogo foram construídas duas classes, uma para guardar a informação referida anteriormente (com o nome **InfoSG**) e outra (**SaveManager**) para guardar esta informação num ficheiro e para voltar a carregar esta informação na classe anterior a partir do ficheiro. Portanto, quando o jogador clica para guardar o jogo, o código vai percorrer as listas necessárias e a matriz que identifica cada objeto no cenário do jogo, para preencher a classe **InfoSG**, depois a classe **SaveManager** trata de guardar a informação contida nesta classe num ficheiro binário (código disponível nos anexos). Escolheu-se usar um ficheiro binário para conter o jogo guardado, para não existir batota no jogo, ou seja, alguém construir uma cidade a partir do ficheiro onde se guarda o jogo. Para carregar o jogo é só reverter o processo, ou seja, tirar a informação do ficheiro binário para a classe **InfoSG** (C# já faz isso), carregar as listas e matrizes usadas no decorrer do jogo, e construir os edifícios e terreno novamente. É de notar que este sistema de guardar a cidade, não guarda os modelos concretos de cada edifício, por exemplo é possível guardar que uma habitação está na posição X;Z e tem altura H, mas se a casa tem 3 janelas e não duas em cada andar, ou não tem telhado, isso já não é possível, pelo menos da maneira como foi implementado. Portanto ao carregar o jogo a cidade no que interessa estará igual, mas os modelos dos edifícios serão quase de certeza diferentes de quando foram guardados.

4.2.9 Tutorial

Ensinar aos jogadores como se joga, foi uma parte difícil de realizar no desenvolvimento. Pois é necessário explicar corretamente como se joga, sem ser muito aborrecido a fazer essa explicação. A ideia inicial, era criar um tutorial interativo, ou seja, indicar ao jogador como se joga e verificar se ele realizou o que foi dito. Esta ideia, é a que a maior parte dos jogos deste tipo faz, mas porque se achou que seria demasiado difícil realizar (num curto espaço de tempo), optou-se por outra estratégia. A primeira foi a realização de texto dinâmico, que muda conforme o menu que o jogador escolhe. Nos primeiros testes em pessoas que se realizou, este método provou ser errado pois havia muito texto, que ninguém queria ler. Sendo assim, criou-se um sistema de vídeo no Jogo. O sistema de vídeo é basicamente um plano que ocupa todo o ecrã do jogo e vai mudando a textura desse plano com as imagens do vídeo escolhido, também se adicionou som e um botão para saltar o vídeo, caso o jogador não o queira ver. Uma parte importante que não se conseguiu fazer foi uma barra para avançar no tempo do vídeo. Este sistema de vídeo teve melhor resultados, mas os vídeos eram muito grandes e detalhados, de modo, que aborreciam o jogador. Por fim, diminui-se o número e tempo de vídeos e mostrou-se apenas o essencial em vez do detalhe. Portanto, na última versão do jogo optou-se por usar

alguns vídeos curtos, só com o essencial, para explicar as partes mais difíceis, nas partes mais fáceis, usou-se texto.

4.2.10 Música

Este tópico é muito aberto a nível de desenvolvimento, como tantos outros, portanto podia-se implementar muitas coisas interessantes nesta área, como usar música dinamicamente conforme a "Saúde da Cidade". Visto que o tempo já era limitado optou-se por uma via mais simples como ter música de fundo a correr no jogo. Para isso, é necessário ter uma lista de músicas e um algoritmo que passe de uma música para a outra, sem repetir sempre a mesma ordem. A parte mais difícil de fazer nesta secção foi a de criar ou escolher a música a usar. A solução que se usou, foi usar um programa online que cria música artificialmente e grátis. No site http://tones.wolfram.com é possível gerar músicas até 30 segundos de forma artificial, para criar as músicas é possível escolher e mudar parâmetros, de modo a produzir músicas diferentes. Para não ter sempre a mesma sequência de música a passar, usou-se o seguinte algoritmo:

```
for (int i = 0; i < musicArray.Length; i++)
{
	do
	{
		i2 = Random.Range(0, musicArray.Length);
	} while (i2 == i);
	a1 = musicArray[i];
	a2 = musicArray[i2];
	musicArray[i] = a2;
	musicArray[i2] = a1;

 }
```

O que este algoritmo faz é percorrer a lista com as músicas e em cada iteração escolhe aleatoriamente outro índice dentro da lista, que seja diferente do índice atual. Quando escolhe o novo índice troca na lista as músicas. Assim sempre que se inicia um novo jogo a ordem das músicas é sempre diferente.

4.3 RESULTADOS E ANÁLISE

4.3.1 Introdução

Nesta secção pretende-se apresentar e analisar os resultados obtidos, sendo eles o questionário que se propõe fazer no jogo e os dados enviados por mail sobre o jogo. A obtenção de resultados para esta dissertação em particular, revelou ser complicada, pois é difícil arranjar pessoas para jogar um jogo e depois responder a um questionário, por isso fez-se o melhor com o que se tem. No fim deste capítulo, serão tiradas algumas conclusões dos resultados obtidos. Depois de concluído o protótipo, colocou-se o jogo na google drive em https://goo.gl/ZfDVbD e no site http://stb.uninova.pt/#projetos para ficar disponível para todos, o executável do jogo. O questionário que se fez, tem como objetivo, verificar se o que se propôs ensinar foi transmitido ou não. O que se propôs ensinar foi o seguinte:

1. O problema que levou à criação do conceito das SC.
2. Ensinar como a obtenção de informação (Big Data, IoT), ou melhor conhecimento, sobre a cidade, através do uso de IoT, pode ajudar a fazer uma gestão mais inteligentes dos recursos da cidade.
3. Explicar a importância do meio ambiente e da educação nas SC.

4.3.2 Resultados

Tendo introduzido este tema passemos para os resultados do questionário.

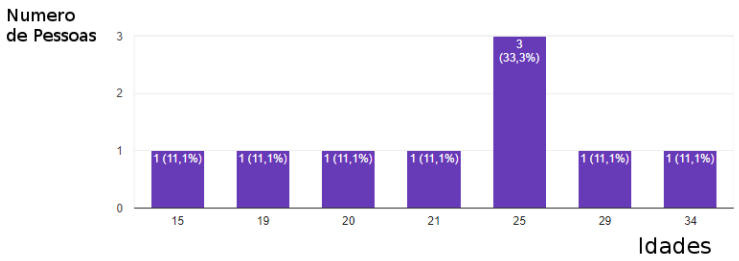


Figura 51 - Pergunta sobre a Idade dos participantes

A partir do gráfico da Figura 51 é possível ver as idades de quem jogou o jogo e respondeu ao formulário. Não se conseguiu resultados com crianças que foi uma pena, a idade mais baixa que conseguiu ter resultados foi de 15 anos, mas já dá para ter uma noção de como se comporta o jogo na área da educação e divertimento/envolvimento.

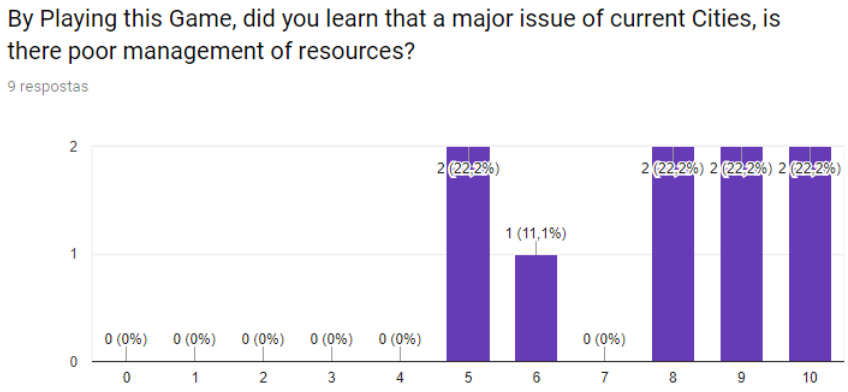


Figura 52 - Segunda pergunta do questionário

Um dos principais problemas que criou o conceito das SC foi o problema da má gestão de recursos da cidade. No gráfico da Figura 52 perguntou-se aos participantes se aprenderam sobre este conceito dando uma avaliação de 0 a 10, onde 0, caso não se tenha aprendido nada do que foi perguntado e 10 caso tenham aprendido. A ferramenta usada para explicar este problema no jogo, foi maioritariamente o vídeo inicial e um pouco da gestão da cidade que se faz no jogo. A média dos resultados é de 7.78, ou seja, as pessoas perceberam relativamente bem este assunto. De facto, apenas 33.3%, das pessoas é que votou abaixo de 7 e ninguém votou abaixo de 5. A única falha é mesmo a faixa etária dos jogadores pois com estes resultados, não é possível entender se uma pessoa mais nova tinha a capacidade de captar esta questão no jogo, apesar de o vídeo inicial ser muito simples e curto.

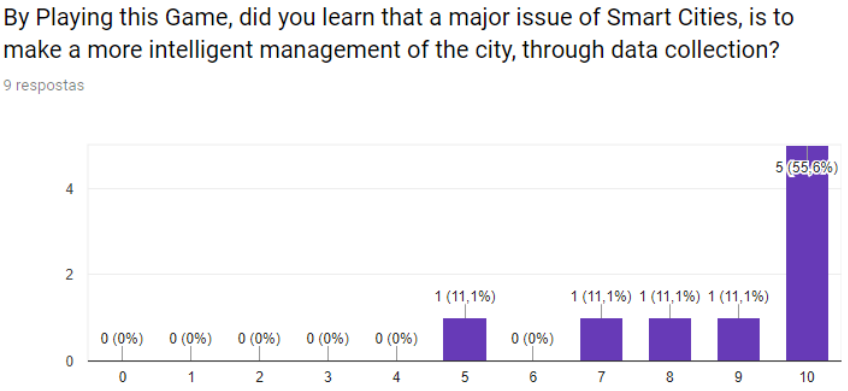


Figura 53 - Terceira pergunta do questionário

Na mesma lógica da pergunta anterior, o gráfico da Figura 53 mostra se os participantes aprenderam sobre o segundo conceito que se pretendia ensinar, na proposta. Para ensinar este conceito (tratado na Figura 53) no jogo, recorreu-se ao vídeo inicial, ao facto de todos os edifícios terem de estar ligados ao EGC e aos textos explicativos por baixo de cada item, no painel/menu de compra de edifícios para a cidade. Os resultados obtidos também foram satisfatórios (tendo uma média de 8.78 foi a pergunta com melhor resultado) a maioria das pessoas entendeu e nenhuma pessoa, tal como na pergunta anterior, respondeu abaixo de 5, portanto este tópico foi bem transmitido no jogo.

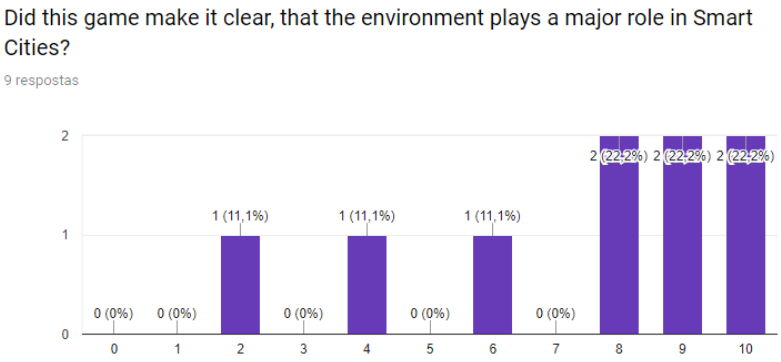


Figura 54 - Quarta pergunta do questionário

Na quarta pergunta do questionário, pretende-se perceber se quem jogou o jogo percebeu que um conceito fundamental nas SC é a preocupação com o meio ambiente. Na mesma lógica das perguntas anteriores, dar valor 0 nesta pergunta significa que o jogo não deu a entender a importância que o meio ambiente tem nas SC e 10 o oposto. Esta questão teve piores resultados do que se esperava, pois achou-se que este tema estaria bastante explícito no jogo, visto que, para não se perder no jogo é preciso criar uma cidade minimamente saudável a nível ambiental. A média dos resultados é de 7.33, ficou um pouco abaixo da expectativa, mas ainda é bastante bom.

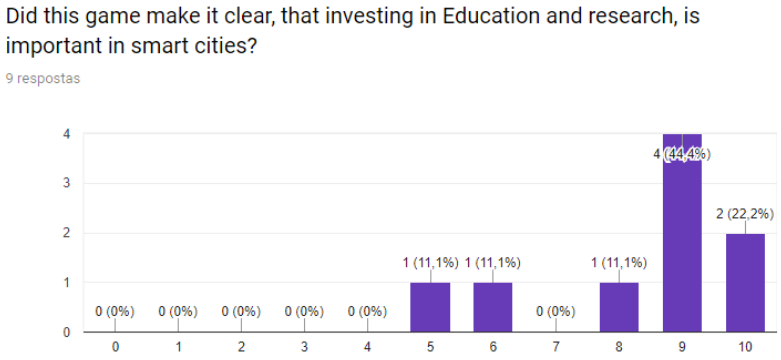


Figura 55 - Quinta pergunta do questionário

Semelhante à pergunta anterior, a pergunta 5 pretende perceber se os jogadores ao jogarem o jogo perceberam a importância da educação nas SC. A avaliação funciona do mesmo modo que na pergunta anterior. A média dos resultados obtidos é a segunda melhor de todas as perguntas e igual a 8.33. São resultados bastantes satisfatórios, mas mais uma vez, não é possível determinar com melhor certeza se em crianças teria os mesmos resultados.

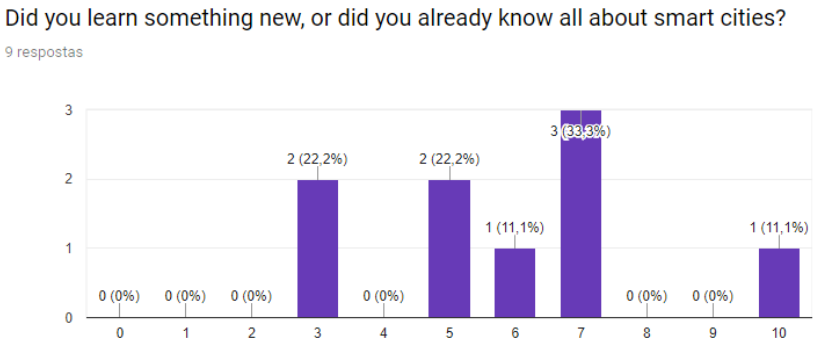


Figura 56 - Sexta pergunta do questionário

Esta pergunta deseja perceber, se os participantes já sabiam muito do tema ou não, pois uma pessoa que já saiba sobre o tema, possivelmente vai captar melhor o que se pretende ensinar do que uma pessoa que não saiba, por exemplo crianças. A avaliação tem a seguinte lógica, 0 caso o jogador já soubesse muito sobre SC e 10 se sabiam pouco e aprendeu muito ao jogar o jogo. A média dos resultados é de 5.89, isto significa que a média dos jogadores já tinha algum conhecimento sobre SC. Como nenhum jovem abaixo dos 15 anos jogou o jogo, os resultados ficaram um pouco abaixo do que o expectável, pois uma criança saberia menos sobre SC, portanto poderia aprender mais com o jogo, do que alguém que já saiba um pouco. Aliás o jogador de 15 anos é que deu 10 valores nesta pergunta.

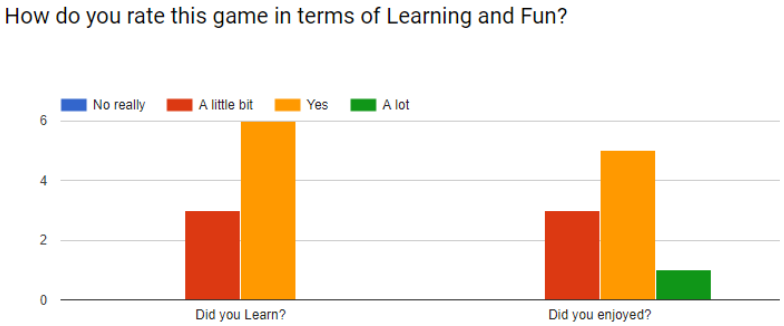


Figura 57 - Sétima e última pergunta obrigatória

Por fim a última pergunta obrigatória do questionário, pois ainda existe uma para o jogador poder dizer o que quiser sobre o jogo (por exemplo dar sugestões), pretende perceber se o jogador gostou de jogar e se aprendeu algo com o jogo ou não. A partir da Figura 57 é possível perceber que tanto a nível de aprendizagem como de divertimento houve resultados medianos.

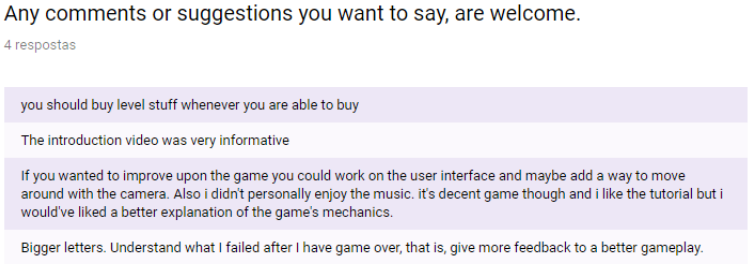


Figura 58 - Última pergunta do questionário, pergunta não obrigatória

A Figura 58 será analisada mais à frente no subcapítulo 4.4, e mostra alguns comentários de algumas pessoas que jogaram o jogo.

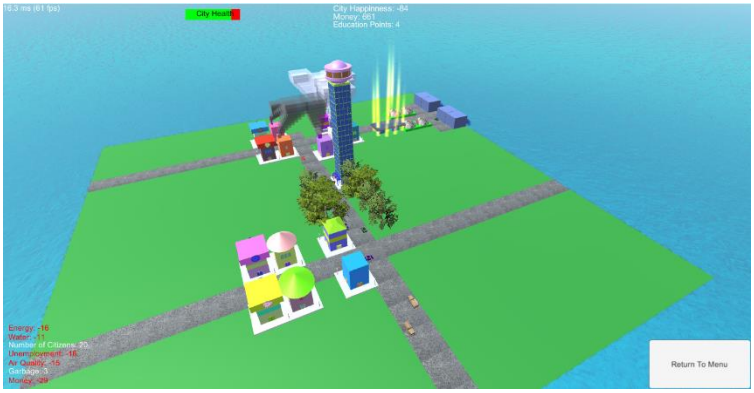


Figura 59 - Uma cidade feita por um jogador

Recebeu-se esta cidade (representada na Figura 59) através do sistema de recolha de dados explicado previamente em 4.1.5. Infelizmente não se conseguiu ter os resultados de todas as pessoas que jogaram, via mail, pois inicialmente só um jogador, que tenha guardado um jogo (uma cidade), é que poderia enviar um mail com o ficheiro da cidade guardada. Como a maior parte dos jogadores, não guardou um jogo, a aplicação não enviou um email. Ao aperceber-se disto, alterou-se a versão do jogo para enviar um email, com a cidade construída até ao momento, ou seja, mesmo que o jogador não tenha guardado o jogo, a aplicação envia um ficheiro com a última cidade construída até ao momento. Mesmo assim não se chegou a receber muitos emails, apenas 1, por problemas desconhecidos, talvez segurança.

Os resultados obtidos pelo questionário, apesar de serem bons e terem relevância para o estudo realizado nesta dissertação, não são os resultados mais desejados, pois, o que mais se pretendia, era testar o jogo em crianças com idades entre 6 – 13 anos de idade, para qual o jogo foi mais direcionado. Felizmente conseguiu-se participar em dois eventos, onde se teve a oportunidade de testar o jogo com crianças (destas idades), que foram a **Noite Europeia dos Investigadores** e **Movimento Código Portugal** ambos no Pavilhão do Conhecimento.

Durante estas experiências não se conseguiu obter resultados quantitativos, como nos questionários feitos, em vez disso conseguiu-se obter resultados qualitativos sobre a aprendizagem e envolvimento que as crianças tiveram no jogo. Não se contou o número de crianças que testaram o jogo, visto que nem sempre jogaram tempo suficiente para extrair alguma informação, mas com certeza 10 ou mais crianças jogaram tempo suficiente para extrair resultados qualitativos, nestes dois eventos.

Resultados Qualitativos Obtidos:

As crianças se envolveram mais com o jogo do que os jogadores que participaram no questionário. Muitas das crianças para além de terem gostado do jogo, ficaram bastante tempo a jogar (alguns chegaram a jogar mais de uma hora). Muitos dos jogadores se preocuparam com a sustentabilidade da cidade, mas o que teve mais sucesso foi a liberdade na criação da cidade, como se pode ver nas próximas duas imagens.

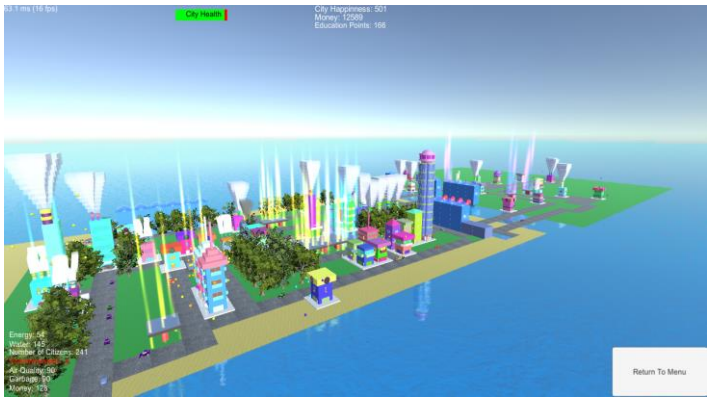


Figura 60 - Cidade Sustentável criada

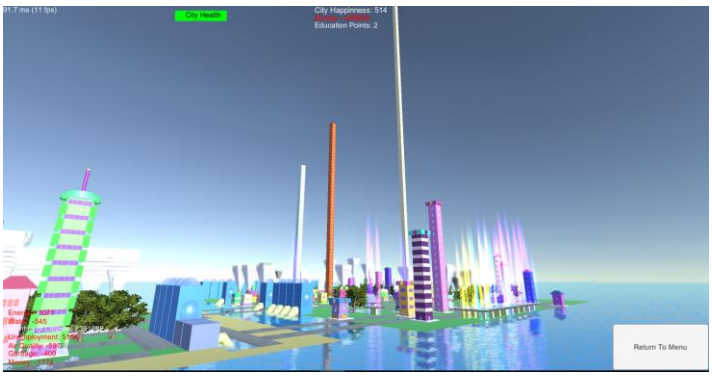


Figura 61 - Cidade Não Sustentável criada

Como se pode ver a Figura 60 tem todos os recursos a branco (positivos), exceto o desemprego (com valor igual a -6 que não é mau para o tamanho da cidade). Na Figura 61, para além de os recursos estarem todos a vermelho, a cidade é claramente um teste às capacidades do jogo (vendo a altura dos prédios), diga-se de passagem, que a jogadora que criou a cidade, só não perdeu o jogo pois parou o tempo de cada turno.

Em relação a resultados na área da aprendizagem. Notou-se que as questões que ficaram mais bem compreendidas pelas crianças, foram as que estavam mais relacionadas com a jogabilidade. Por exemplo as crianças ligaram à questão da sustentabilidade nas Cidades Inteligentes porque, caso não o fizessem perderiam o jogo. Em relação à importância do meio ambiente e da educação, também teve algum sucesso visto que eram necessários para a cidade crescer e sobreviver. Outro ensinamento que se pretendia transmitir e que foi mais ou menos percebido pelas crianças foi a recolha de dados. Tentou-se ensinar este assunto através do vídeo inicial, a conexão dos edifícios ao EGC e através de alguns textos espalhados pelos menus do jogo. Mesmo assim notou-se que faltou o uso de algum tipo de dica visual que realça-se a recolha de dados feita pelo EGC. O vídeo inicial apesar de estar bem feito para jovens/adultos como referido na Figura 58, não teve muito sucesso com as crianças. Por isso, todos os conhecimentos que o vídeo pretendia transmitir ficaram sem efeito.

4.4 VALIDAÇÃO DE REQUISITOS

No capítulo 3.6, foram determinados alguns requisitos educacionais e lúdicos a cumprir. Da parte educacional, pelos resultados obtidos, compreendeu-se que conseguiu-se passar os conhecimentos propostos. Mesmo nas crianças, que tiveram pior performance nesta questão comparativamente aos jovens/adultos, também chegaram a aprender suficiente, como foi referido em 4.3.2 . Nas questões lúdicas, também se conseguiu cumprir os requisitos colocados no capítulo 3.6, exceto na questão da rápida curva de aprendizagem. Este requisito é claramente muito importante, visto que influencia outros requisitos funcionais. Como foi referido, as crianças por terem tido uma pessoa a explicar como se jogava, conseguiram se envolver mais com o jogo, do que os jovens/adultos. Concluindo, obteve-se sucesso a cumprir quase todos os requisitos propostos, menos na rápida curva de aprendizagem, onde se notou que sem uma ajuda externa (alguém que já saiba jogar), jogar o jogo, torna-se bastante difícil principalmente para compreender como se cria uma cidade sustentável no jogo.

5 CONCLUSÃO

O objetivo desta dissertação foi o de estudar a performance dos jogos digitais a ensinar os conceitos básicos e fundamentais das **SC**. Considerou-se os jogos digitais uma boa ferramenta de ensino, pois são mais cativantes para os jovens/crianças e muito envolventes, o que faz com que um indivíduo dedique tempo ao tema das **SC**, portanto aprende sobre o assunto.

Os jogos sérios, têm um grande obstáculo para o seu sucesso. Ter sucesso no contexto dos jogos sérios é no fundo ensinar o melhor possível, sendo o menos aborrecido possível. Para isso é preciso gerir muito bem a parte lúdica do jogo com a educativa. Este foi, o grande obstáculo sentido na dissertação e também em geral na criação de jogos sérios.

Pelos resultados obtidos nos questionários, percebeu-se que se estava a conseguir passar as ideias fundamentais propostas. A maior parte dos resultados obtidos, foram positivos, sendo a média dos resultados médios de cada pergunta igual a 7.62, o que é bom visto que o questionário foi feito de modo a que quanto maior (até 10) fosse a nota dada em cada pergunta, melhor o resultado do protótipo feito. Apesar de a amostra de pessoas que jogaram e responderam ao questionário ser muito pequena e não ter a faixa etária que se pretendia estudar mais, pode-se concluir, dentro do possível, que o protótipo usado ensinou o que se pretendia, sendo que 66.7% dos jogadores disse que aprendeu, ao jogar o jogo. Em relação à componente lúdica do jogo, obteve-se resultados mistos, pois por uma lado no questionário ao perguntar se as pessoas gostaram do jogo, apenas 33.3% dos jogadores (3 em 9) disse que tinha gostado pouco do jogo, mas por outro lado foi muito difícil encontrar pessoas que quisessem jogar o jogo (tendo partilhado o jogo no reddit numa secção própria para isso, o jogo não teve muito interesse por parte da comunidade, daí os poucos resultados obtidos), também se notou que houve falta de envolvimento por parte dos jogadores pois poucos chegaram a fazer uma cidade sustentável. Por outro lado, como se pode ver na Figura 59 um jogador chegou a fazer uma cidade minimamente avançada, visto que tinha já algumas fábricas de nível 2, que ainda leva algum tempo.

Portanto, pelas pessoas que jogaram e responderam ao questionário percebeu-se que os jogadores aprenderam os conhecimentos que o jogo pretendia transmitir. Em relação ao envolvimento com o jogo, que também é importante para que a transmissão de conhecimento funcione bem num jogo sério, já se obteve resultados piores, que se devem à má implementação do tutorial. Como se pôde ver na Figura 58, a maior parte dos comentários têm a ver com a interface do jogo, que não ficou muito fácil de usar, tornando assim o jogo mais difícil de jogar. Este facto ainda se tornou mais patente, depois de se ter testado o jogo com as crianças nos dois eventos mencionados previamente. Como é que as crianças conseguiram criar cidades sustentáveis e os mais velhos que responderam ao questionário não? Isto aconteceu porque as crianças tiveram uma pessoa a servir de tutorial do jogo, enquanto que os jogadores que responderam ao questionário não. Mas tendo está parte (tutorial) bem-feita, verificou-se que o jogo teve sucesso, principalmente com as crianças pelo menos na parte do entretenimento. Em relação à aprendizagem o jogo, teve piores resultados com crianças do que nos adultos, mas ainda assim, bastante satisfatórios nas áreas mencionadas no capítulo anterior.

5.1 TRABALHOS FUTUROS

Neste subcapítulo pretende-se listar algumas melhorias que se podiam fazer ao jogo de modo a obter mais sucesso tanto na parte educativa como lúdica.

A parte mais frustrante para os jogadores, foi a parte tutorial. Deveria ter sido realizado, como foi descrito na proposta, e como a maior parte dos jogos de simulação de cidades funcionam, que é no fundo mostrar passo a passo como funciona o jogo, deixando o jogador fazer cada um desse passos. Assim tornava-se o jogo, desde o começo mais interativo.

Outra dificuldade que se percebeu foi o painel de compra de edifícios para a cidade. Sendo uma das ferramentas essenciais para jogar, estava demasiado complexo, pois tinha desde o começo todos os objetos disponíveis sendo que alguns não davam para comprar. Isto criou confusão, e alguma frustração ao jogador, pois não é intuitivo quais os objetos que o jogador pode comprar ou não. Para melhorar esta parte o jogador deveria ter acesso a menus hierárquicos, ou seja, o jogador devia ter acesso a mais submenus dentro de cada menu. Outro problema relacionado com o painel é que cada botão mostra muita informação e deveria ser mais focado, por exemplo, só mostrar o que cada bloco produz e consome, visto que a maior parte dos jogadores não lê sequer este texto.

Uma questão que poderia melhorar bastante, é o de explicar a **recolha de dados** feita pelo **Edifício Gestor da Cidade**. Por exemplo, todos os edifícios deveriam emitir algum tipo de partículas até ao EGC de modo a evidenciar a comunicação que existe entre os elementos da cidade e o EGC.

Para facilitar a experiência do jogo, também era interessante mostrar ao jogador automaticamente, quando um elemento não está conectado ao EGC. Atualmente, o jogador só consegue perceber isso caso verifique visualmente que um elemento não tem nenhuma estrada ao seu lado ou em caso de dúvida, ligando o modo de conexão que indica os elementos ligados ao EGC.

Uma das reclamações na parte lúdica do jogo, que se obteve por parte das crianças foi que as praias que o jogador pode construir, são só decorativas e não têm pessoas. Pegando nesta reclamação podemos inferir que uma melhoria a realizar no jogo seria dar mais movimento/vida à cidade. Por exemplo, colocando pessoas a jogar na praia, ou peões nas estradas, ou ainda painéis solares que rodem com o sol, seriam boas adições ao jogo.

Outra melhoria a fazer seria dar importância à localização geográfica de cada edifício, por exemplo, era interessante que as lixeiras e fábricas tivessem longe das habitações. Para fazer isto bastava usar a mecânica de conexão dos edifícios ao EGC da seguinte forma: as habitações deveriam ficar perto do EGC e os edifícios poluentes longe do EGC, isto é relativamente simples de fazer, pois todas as estradas sabem a que distância estão do EGC. A partir de aqui era apenas necessário arranjar uma forma de beneficiar o jogador que tivesse esta preocupação.

Seria muito interessante, concluir a versão móvel do jogo, que podia ser mais cativante para o jogador de hoje em dia, mais habituado a jogar jogos em dispositivos móveis.

Outra melhoria importante seria a realização de uma versão do jogo em português.

Por fim, uma outra ideia, para um futuro trabalho, seria alterar a ideia base do jogo, tentar ir por um caminho diferente do que foi dado nesta dissertação. Simuladores de cidades já não são uma ideia original, de facto já existe muitos concorrentes (não aplicadas no mesmo contexto

desta dissertação), uma nova ideia de jogo sobre Cidades inteligentes que ensine e seja criativo de modo a ter uma implementação possível de realizar e que envolva os jogadores, poderia ter bons resultados, o problema seria encontrar essa ideia.

6 REFERÊNCIAS

[1] K. Squire, “Video Games in Education,” *Int. J. Intell. Simulations Gaming*, vol. 2, pp. 49--62, 2003.

[2] G. Learning, “Digital Games Revolutionizing Workplace Learning?”

[3] United Nations, *World Urbanization Prospects: The 2014 Revision, Highlights (ST/ESA/SER.A/352)*. 2014.

[4] United Nations, “The World’s Cities in 2016 – Data Booklet (ST/ESA/ SER.A/392),” in *The World’s Cities in 2016*, 2016.

[5] United Nation, “UN finds world's population is increasingly urban with more than half living in urban areas today and another 2 . 5 billion expected by 2050,” *Dep. Econ. Soc. Aff.*, pp. 2014–2016, 2014.

[6] “How It Works: Smarter Cities - YouTube.” [Online]. Available: https://www.youtube.com/watch?v=yJVK25wWvbE. [Accessed: 18-Jan-2017].

[7] D. (David R. . Michael and S. L. Chen, *Serious games : games that educate, train and inform*. Itps Thomson Learning, 2006.

[8] B. Gros, “Game Dimensions and Pedagogical Dimension in Serious Games,” *Handb. Res. Serious Games Educ. Appl.*, p. 402, 2016.

[9] D. Djaouti, J. Alvarez, and J.-P. Jessel, “Classifying Serious Games,” pp. 118–136.

[10] “About - IEEE smart cities.” 2016.

[11] “What are Smart Cities? | Larissa Suzuki | TEDxUCLWomen - YouTube.” [Online]. Available: https://www.youtube.com/watch?v=Kqkoghq0G4A. [Accessed: 26-Jan-2017].

[12] “KILLER AIR Berkeley Earth Publishes Study on Air Pollution in China.”

[13] D. Hoornweg and P. Bhada-Tata, “Urban Development - What a Waste: A Global Review of Solid Waste Management,” 2012.

[14] D. for B. I. & S. Uk, “SMART CITIES: Background paper,” no. October, p. 47, 2013.

[15] J. Ramalho, “Faculdade de Engenharia da Universidade do Porto,” FEUP, 2015.

[16] C. Manville, R. Europe, J. Millard, D. Technological Institute, A. Liebe, and R. Massink, “Mapping Smart cities in the EU.”

[17] “How we design and build a smart city and nation | Cheong Koon Hean | TEDxSingapore - YouTube.” [Online]. Available: https://www.youtube.com/watch?v=m45SshJqOP4. [Accessed: 10-Feb-2017].

[18] “Press Pictures,” *siemens*, 2012. [Online]. Available: http://www.siemens.com/press/en/presspicture/?press=/en/presspicture/2011/mobility-logistics/soicmol201103/soicmol201103-04.htm&content[]=ICMOL&content[]=MO. [Accessed: 17-Feb-2017].

[19] “Smart cities are about people not technology – video | Public Leaders Network | The

Guardian.” [Online]. Available: https://www.theguardian.com/public-leaders-network/video/2015/aug/03/smart-cities-are-about-people-not-technology-video. [Accessed: 26-Jan-2017].

[20] “A smart city: It’s not just about technology | Diane Wang | Pulse | LinkedIn.” [Online]. Available: https://www.linkedin.com/pulse/smart-city-its-just-technology-diane-wang?articleId=6174405929858101248. [Accessed: 28-Jan-2017].

[21] G. S. Barbosa, P. Drach, and O. D. Corbella, “Análise do Projeto Urbano de Masdar a Partir de Categorias Analíticas de Projetos Sustentáveis,” *An. do Encontro Latinoam. Edif. e Comunidades Sustentáveis - 2013*, no. January, 2013.

[22] “Masdar-designed smart villas to cut household energy use | The National.” [Online]. Available: http://www.thenational.ae/uae/environment/masdar-designed-smart-villas-to-cut-household-energy-use. [Accessed: 17-Feb-2017].

[23] “Masdar City to test GE ‘smart’ appliances - CNET.” [Online]. Available: https://www.cnet.com/news/masdar-city-to-test-ge-smart-appliances/. [Accessed: 17-Feb-2017].

[24] “Welcome To Masdar City - YouTube.” [Online]. Available: https://www.youtube.com/watch?v=FyghLnbp20U. [Accessed: 03-Feb-2017].

[25] “Is this the greenest city in the world? - YouTube.” [Online]. Available: https://www.youtube.com/watch?v=w2pRGuY0WxE. [Accessed: 03-Feb-2017].

[26] “Masdar City Welcome Video (3 minutes, English) - YouTube.” [Online]. Available: https://www.youtube.com/watch?v=Llzq9YMsPP8. [Accessed: 03-Feb-2017].

[27] “Portal das Finanças,” *Portal*, p. 2011, 2011.

[28] Agência para a Modernização Administrativa, “Portal do Cidadão.” [Online]. Available: https://www.portaldocidadao.pt/. [Accessed: 02-Mar-2017].

[29] “e-Governance - City of Stockholm.” [Online]. Available: http://international.stockholm.se/governance/e-governance/. [Accessed: 05-Feb-2017].

[30] “E-sErvicEs in stockholm.”

[31] “Addressing Solar Intermittency with a Home Energy Management System (HEMS) - YouTube.” [Online]. Available: https://www.youtube.com/watch?v=bNsN6sZUlr8. [Accessed: 06-Feb-2017].

[32] “Smart grid brings U.S. power into 21st century | Environmental Defense Fund.” [Online]. Available: https://www.edf.org/climate/smart-grid-brings-us-power-21st-century?utm_source=ggad&utm_medium=cpc&utm_campaign=gr-PecanStreet&gclid=COXa-be9qNACFUu3Gwodwy0D3A. [Accessed: 06-Feb-2017].

[33] “Tesla Unveils Powerwall 2 & Solar Roof - YouTube.” [Online]. Available: https://www.youtube.com/watch?v=4sfwDyiPTdU. [Accessed: 17-Feb-2017].

[34] “Solar – Solar Energy Efficiency | SolarCity.” [Online]. Available: http://www.solarcity.com/. [Accessed: 17-Feb-2017].

[35] Amazon, “Amazon Go,” 2016. [Online]. Available: https://www.amazon.com/b?node=16008589011. [Accessed: 02-Mar-2017].

[36] “Lidl Shop&Go - Lidl Portugal.” [Online]. Available: http://www.lidl.pt/pt/lidl-shop-go.htm. [Accessed: 02-Mar-2017].

[37] “Amazon Prime Air - YouTube.” [Online]. Available: https://www.youtube.com/watch?v=MXo_d6tNWuY. [Accessed: 06-Feb-2017].

[38] “TU Delft - Ambulance Drone - YouTube.” [Online]. Available: https://www.youtube.com/watch?v=y-rEI4bezWc. [Accessed: 06-Feb-2017].

[39] “Dubai announces passenger drone plans - BBC News.” [Online]. Available: http://www.bbc.com/news/technology-38967235. [Accessed: 17-Feb-2017].

[40] “Augmented Reality | Stuttgart - Smart Cities - Ready to go! - Solid White - YouTube.” [Online]. Available: https://www.youtube.com/watch?v=WdhK1Km2Cw0. [Accessed: 02-Mar-2017].

[41] “Smart City (Serious Game).” [Online]. Available: https://preview.c9users.io/blinkst/phaser_demo/index.html?_c9_id=livepreview0&_c9_host=https://ide.c9.io. [Accessed: 02-Mar-2017].

[42] “Block’hood Home Page.” [Online]. Available: http://www.plethora-project.com/blockhood/. [Accessed: 01-Mar-2017].

[43] “SimCity Hotels Block.png (1920×1080).” [Online]. Available: http://web-vassets.ea.com/Assets/Resources/Image/Screenshots/SimCity Hotels Block.png?cb=1412974396. [Accessed: 01-Mar-2017].

[44] “SimCity: Digestible Complexity - IGN.” [Online]. Available: http://www.ign.com/articles/2012/08/15/simcity-digestible-complexity. [Accessed: 01-Mar-2017].

ANEXOS

CÓDIGO RESPONSÁVEL POR DETETAR ONDE O JOGADOR CLICOU NO MAPA DO JOGO

```
// Update is called once per frame
    void Update () {

        if (Input.GetMouseButtonDown(0)&& menu.touchGameArea())
        { // if left button pressed...
            Ray ray = camera.ScreenPointToRay(Input.mousePosition);
            RaycastHit hit;
            m.enabled = true;
            if (Physics.Raycast(ray, out hit))
            {

                // the object identified by hit.transform was clicked
                // do whatever you want
                Vector3 picked_place = hit.transform.position;
                picked_place.y = 1;
                this.transform.position = new
Vector3(/*(int)*/picked_place.x,picked_place.y,/*(int)*/picked_place.z
);

            }

        }
        if (m.enabled && !Input.GetMouseButton(0) &&
!menu.edit_select)
        {
            m.enabled = false;
        }

    }
```

CÓDIGO PARA COLOCAR JANELAS NO EDIFÍCIO

```
void CreateWindows(Building parent, int height, float width, GameObject[] windows)
    {

        GameObject window, _window;
        Transform obj;
        Vector3 pos = parent.parentGameObject.transform.position;
        Color windowColor = new Color(Random.value, Random.value, Random.value, 1.0f);
        Color windowColor2 = new Color(Random.value, Random.value, Random.value, 1.0f);
        Color windowColor3 = new Color(Random.value, Random.value, Random.value, 1.0f);
        Color windowColor4 = new Color(Random.value, Random.value, Random.value, 1.0f);
        float building_measures = width;
        int windowType = Random.Range(0, windows.Length);

        float sd = building_measures / 2;
        sd += 0.01f;
        float building_floorHeight = 0.5f;
        float magicNumber = 0.1f;

        window = Object.Instantiate(windows[windowType]);

        for (int i = 0; i < window.gameObject.transform.childCount; i++)
        {

            obj = window.gameObject.transform.GetChild(i);
            if (obj.tag.Equals("window"))
            {
                obj.GetComponent<Renderer>().sharedMaterial.color = windowColor2;
                SetBrightness2(Brightness, windowColor2, obj);
            }
            else
            {
                if (obj.tag.Equals("armature"))
                {
                    obj.GetComponent<Renderer>().sharedMaterial.color = windowColor3;
                    SetBrightness2(Brightness, windowColor3, obj);
                }
                else
                {
                    if (obj.tag.Equals("armature2"))
                    {
                        obj.GetComponent<Renderer>().sharedMaterial.color = windowColor4;
                        SetBrightness2(Brightness, windowColor4, obj);
                    }
                    else
                    {
                        obj.GetComponent<Renderer>().sharedMaterial.color = windowColor;
                        SetBrightness2(Brightness, windowColor, obj);
                    }
                }
            }


        }
        window.transform.localScale = new Vector3(0.1f, 0.1f, 0.1f);




        for (int i = 0; i < height; i++)
        {

            //front window
            _window = Object.Instantiate(window);
            //colocar janelas na classe building
            for (int j = 0; j < _window.gameObject.transform.childCount; j++)
            {

                obj = _window.gameObject.transform.GetChild(j);
                if (obj.tag.Equals("window"))
                {
                    parent.windows.Add(obj.gameObject);
                }
                else
                {
                    parent.windows_armature.Add(obj.gameObject);
                }

            }
            _window.transform.SetParent(parent.parentGameObject.transform);
            _window.transform.position = new Vector3(pos.x, pos.y + i * building_floorHeight +
magicNumber, pos.z - sd);
            //_window.GetComponent<Renderer>().material.color = windowColor;
```

```
            //SetBrightness(Brightness, windowColor, _window);

            //right window

            _window = Object.Instantiate(window);
            //colocar janelas na classe building
            for (int j = 0; j < _window.gameObject.transform.childCount; j++)
            {

                obj = _window.gameObject.transform.GetChild(j);
                if (obj.tag.Equals("window"))
                {
                    parent.windows.Add(obj.gameObject);
                }
                else
                {
                    parent.windows_armature.Add(obj.gameObject);
                }

            }
            _window.transform.SetParent(parent.parentGameObject.transform);
            _window.transform.position = new Vector3(pos.x + sd, pos.y + i * building_floorHeight +
magicNumber, pos.z);
            _window.transform.localEulerAngles = new Vector3(0, -90, 0);

            //left window

            _window = Object.Instantiate(window);
            //colocar janelas na classe building
            for (int j = 0; j < _window.gameObject.transform.childCount; j++)
            {

                obj = _window.gameObject.transform.GetChild(j);
                if (obj.tag.Equals("window"))
                {
                    parent.windows.Add(obj.gameObject);
                }
                else
                {
                    parent.windows_armature.Add(obj.gameObject);
                }

            }
            _window.transform.SetParent(parent.parentGameObject.transform);
            _window.transform.position = new Vector3(pos.x - sd, pos.y + i * building_floorHeight +
magicNumber, pos.z);
            _window.transform.localEulerAngles = new Vector3(0, 90, 0);

            //back window

            _window = Object.Instantiate(window);
            //colocar janelas na classe building
            for (int j = 0; j < _window.gameObject.transform.childCount; j++)
            {

                obj = _window.gameObject.transform.GetChild(j);
                if (obj.tag.Equals("window"))
                {
                    parent.windows.Add(obj.gameObject);
                }
                else
                {
                    parent.windows_armature.Add(obj.gameObject);
                }

            }
            _window.transform.SetParent(parent.parentGameObject.transform);
            _window.transform.position = new Vector3(pos.x, pos.y + i * building_floorHeight +
magicNumber, pos.z + sd);
            _window.transform.localEulerAngles = new Vector3(0, -180, 0);


        }
        Object.Destroy(window);

    }
```

CLASSE SAVEMANAGER

```
public class SaveManager
{

    public void SaveGame(InfoSG _info)
    {
        BinaryFormatter bf = new BinaryFormatter();
        FileStream stream = new FileStream(Application.persistentDataPath +
"/smart_city_game.sav", FileMode.Create);
        File.WriteAllText(Application.persistentDataPath + "/smart city game.txt",
_info.WriteData());
        InfoSG data = _info;
        bf.Serialize(stream, data);
        stream.Close();

        //mono gmail g = new mono gmail();
        //g.SendMailWithAttachment();
    }

    public InfoSG LoadGame()
    {
        InfoSG i = null;
        if (File.Exists(Application.persistentDataPath + "/smart_city_game.sav"))
        {
            BinaryFormatter bf = new BinaryFormatter();
            FileStream stream = new FileStream(Application.persistentDataPath +
"/smart_city_game.sav", FileMode.Open);
            i = bf.Deserialize(stream) as InfoSG;
            stream.Close();
        }

        return i;

    }

    public bool checkIfAsDataSave()
    {
        return File.Exists(Application.persistentDataPath + "/smart_city_game.sav");
    }
}

[Serializable]
public class BlockInfoSG
{
    public int blocktype;
    public string description;
    public int height;
    public int x, z;

    public BlockInfoSG(int type, string dis, int h, int posx, int posz)
    {
        x = posx;
        z = posz;
        height = h;
        description = dis;
        blocktype = type;
    }

}

[Serializable]
public class InfoSG
{
    // por tudo a guardar
    public BlockInfoSG[,] city_info;
    public int money;
    public int last_happinnes, last_number_citizens, last_energy, last_water,
last_garbage_collection, last_unemployment, last_air_quality, last_money_perturn;
    public int health;
    public int last_education;
    public int health bar size;
    public int education_cost_lv2, education_cost_lv3;
    //GameObject cube;

    public InfoSG(BlockInfoSG[,] city, int _money)
```

```
    {
        city_info = city;
        money = _money;

    }
    public void PrintData()
    {
        Debug.Log("Money:    " + money + "\n\n\n\n\n");

        foreach (BlockInfoSG b in city info)
        {
            Debug.Log(b.description);
        }

    }

    public void SaveTextInfo(int last_happinnes, int last_number_citizens, int
last_energy, int last_water, int last_garbage_collection, int last_unemployment, int
last_air_quality, int last_money_perturn)
    {
        this.last_happinnes = last_happinnes;
        this.last_number_citizens = last_number_citizens;
        this.last_energy = last_energy;
        this.last_water = last_water;
        this.last_garbage_collection = last_garbage_collection;
        this.last_unemployment = last_unemployment;
        this.last_air_quality = last_air_quality;
        this.last_money_perturn = last_money_perturn;
    }
    //public string WriteData()
    //{
    //    String s;
    //    s = "Money: " + money+"\n";
    //    foreach(BlockInfoSG b in city_info)
    //    {
    //        s += b.description+"\n";
    //    }
    //    return s;
    //}

    public string WriteData()
    {
        String s;
        s = "Money " + money + "\n";
        s += "Happiness " + last_happinnes + "\n";
        s += "Number_of_citizens " + last_number_citizens + "\n";
        s += "Energy " + last_energy + "\n";
        s += "Water " + last_water + "\n";
        s += "Garbage " + last_garbage_collection + "\n";
        s += "Unemployment " + last_unemployment + "\n";
        s += "Air_quality " + last_air_quality + "\n";
        s += "Money_per_turn " + last_money_perturn + "\n";
        s += "Health " + health + "\n";
        s += "Education " + last_education + "\n";
        return s;
    }
}
```

CÓDIGO SOBRE MOVIMENTO DE PARTÍCULAS DE FUMO

```
// Update is called once per frame
    void Update () {
        time += Time.deltaTime;
        time2 += Time.deltaTime;

        if (particles.Count < number_of_particles)
        {
            if (time>creation_interval) {
                  GameObject p = Instantiate(particle,
                  this.transform.position,
                  this.transform.localRotation) as GameObject;

                  Smoke s = new Smoke(p);
                  particles.Add(s);
                  time = 0;
            }
        }

        for(int i=0;i<particles.Count;i++)
        {
            if (time2 > change_direction)
            {
                direction = Random.Range(-0.002f,0.002f);
                time2 = 0;
            }

            Smoke smoke = particles[i];
            if (smoke.duration >= maxduration)
            {
                Destroy(smoke.particle);
                particles.RemoveAt(i);
            }
            else
            {
                Vector3 pos = smoke.particle.transform.position;
                Vector3 scale = smoke.particle.transform.localScale;

                smoke.particle.transform.position = new
                Vector3(pos.x+direction/**speed*/, pos.y +
                Time.deltaTime * speed, pos.z+direction/**speed*/);

                smoke.particle.transform.localScale = new
                Vector3(scale.x+Time.deltaTime*speed/2, scale.y +
                Time.deltaTime * speed/8, scale.z + Time.deltaTime *
                speed/2);

                smoke.duration += Time.deltaTime;
            }
        }

    }
```

CÓDIGO SOBRE EFEITO DE PARTÍCULAS DA FIGURA 38

```
// Update is called once per frame
    void Update ()
    {
         time += Time.deltaTime;

        shape1.transform.localScale = new Vector3(particle_size, particle_size,
        particle_size);
        shape2.transform.localScale = new Vector3(particle_size, particle_size,
        particle size);
        shape1.GetComponent<Renderer>().material.color = color1;
        shape2.GetComponent<Renderer>().material.color = color2;

        sizex = this.transform.localScale.x;
        sizey = this.transform.localScale.y;
        sizez = this.transform.localScale.z;

        x = transform.position.x;
        y = transform.position.y;
        z = transform.position.z;

        float sx = Random.Range(-range, range), sy = Random.Range(-range, range), sz =
        Random.Range(-range, range);
        //Debug.Log(time);
        if (particles.Count < n_particles )
        {
                float n = Random.value;
                GameObject p;
                if (n <= 0.5)
                {
                    p = Instantiate(shape1);

                }
                else
                {
                    p = Instantiate(shape2);

                }

                p.transform.position = new Vector3(x + sx, y + sy, z + sz);

                particles.Add(p);

            }

        if (particles.Count > 0)
        {
            for(int i=0;i<particles.Count;i++)
            {
               GameObject p = particles[i];
               px = p.transform.position.x; py = p.transform.position.y;
               pz = p.transform.position.z;
               float diffx = x - px, diffy = y - py, diffz = z - pz;

               p.transform.position = new Vector3(px + diffx * Time.deltaTime * speed,
               py + diffy * Time.deltaTime * speed, pz + diffz * Time.deltaTime *
               speed);

               if (Mathf.Abs(diffx) < (sizex / 2) && Mathf.Abs(diffy)< (sizey / 2) &&
               Mathf.Abs(diffz) < (sizez / 2))
                {
                    Destroy(p);
                    particles.RemoveAt(i);

                }
            }
        }
    }
```

CÓDIGO MOVIMENTO DE EXTRATOR DE AGUA

```
// Update is called once per frame
void Update () {
        sx = transform.localScale.x;
        sy = transform.localScale.y;
        sz = transform.localScale.z;
        time += Time.deltaTime;
        if (stretch)
        {
            if (time > switchTime)
            {
                time = 0;
                stretch = false;
            }
            this.transform.localScale = new
Vector3(sx+Time.deltaTime*(-speed*factor),sy+Time.deltaTime*speed,sz +
Time.deltaTime * (-speed*factor));
        }
        else
        {
            if (time > switchTime)
            {
                time = 0;
                stretch = true;
            }
            this.transform.localScale = new Vector3(sx +
Time.deltaTime * (speed*factor), sy + Time.deltaTime * (-speed), sz +
Time.deltaTime * (speed*factor));
        }
    }
```